|  |
| --- |
|  |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования  **«МИРЭА - Российский технологический университет»**  **РТУ МИРЭА** |
| Институт информационных технологий (ИТ) |
| Кафедра инструментального и прикладного программного обеспечения (ИиППО) |

|  |  |
| --- | --- |
| **ОТЧЁТ**  **ПО ПРАКТИЧЕСКИМ РАБОТАМ** | |
| **по дисциплине** |  |
| **«Программирование на языке Джава»** | |
| Выполнил студент группы ИМБО-01-21 | Рутковская А.А. |
| Принял ассистент кафедры ИиППО | Ермаков С. Р. |

|  |  |  |
| --- | --- | --- |
| Практические работы выполнены | « » 2022 г. |  |
| «Зачтено» | « » 2022 г. |  |

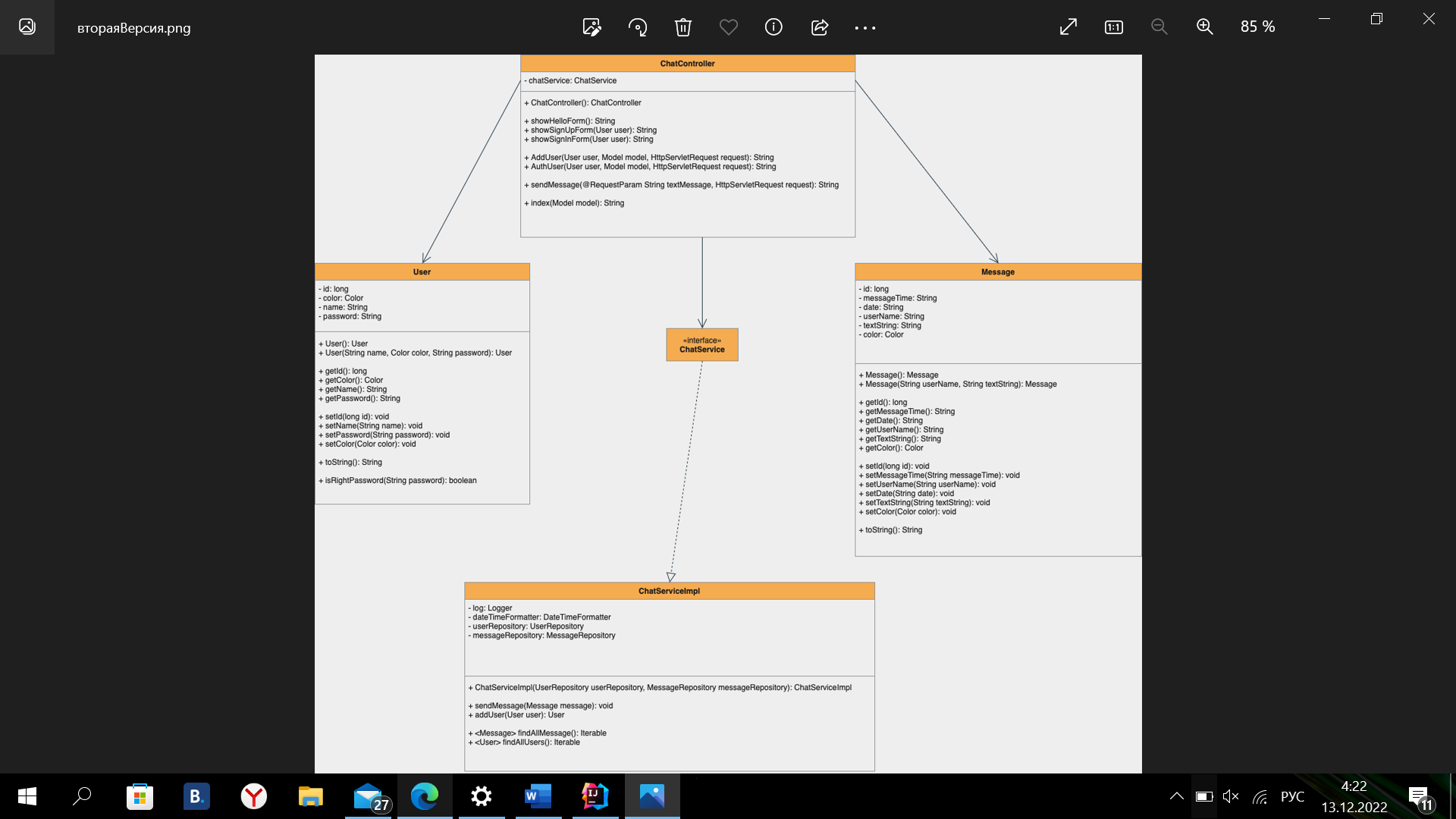
Москва, 2022

# Проектная работа

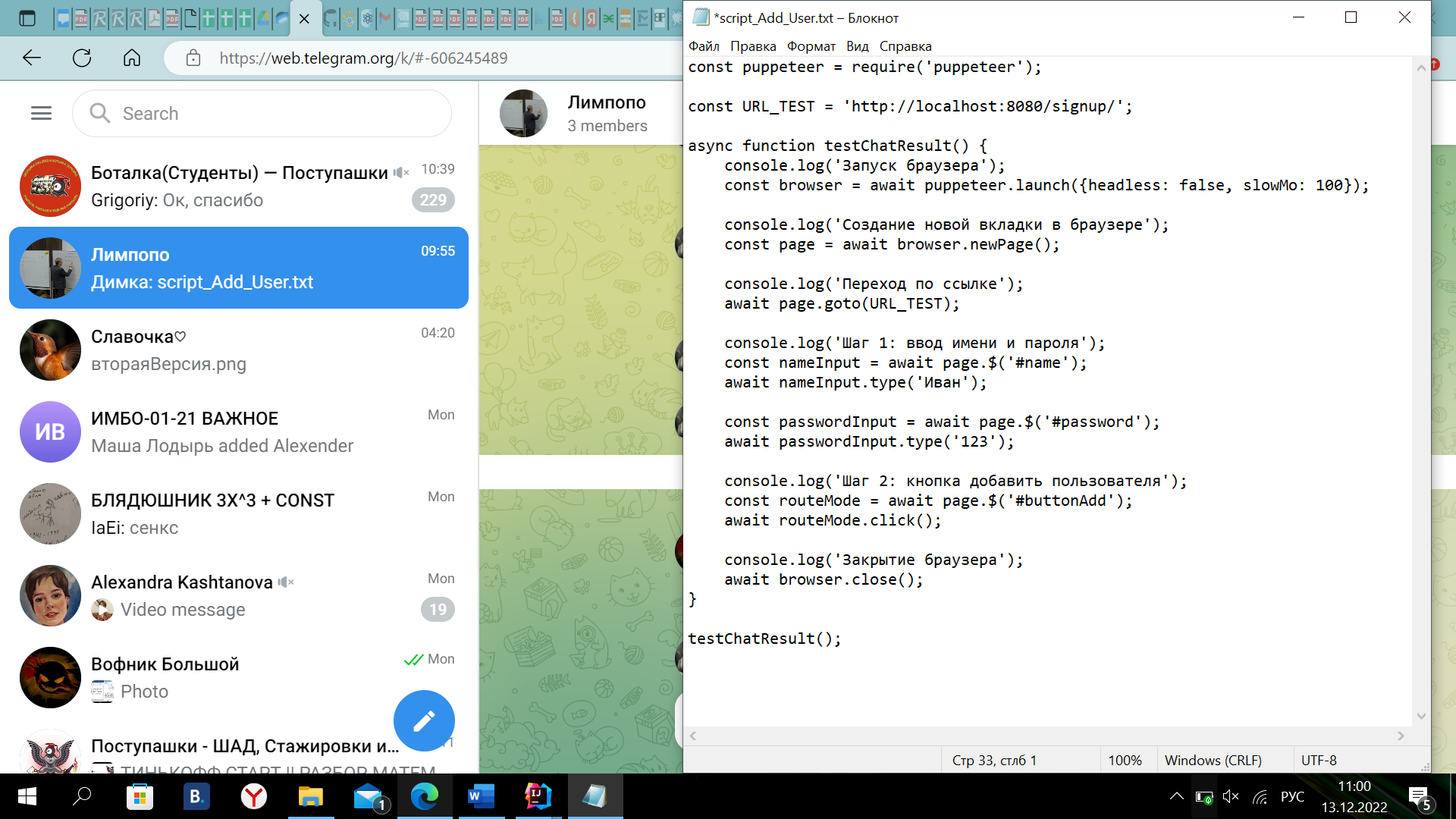
Цель проекта: освоить темы дисциплины «Программирование на языке Джава» путем командной проектной деятельности. В качестве темы проекта была выбрана реализация Web-chat’а. Данный опыт помогает понять как работает frontend и backend программирование.

## Реализованные в проекте темы практических работ

|  |  |  |
| --- | --- | --- |
| **Номер темы** | **Название темы** | **Комментарий студента** |
| 1 | Знакомство со средой разработки. Синтаксис и основные управляющие конструкции языка Джава | Среда разработки – IntelliJ IDEA  Синтаксис языка:   * Чувствительность к регистру – Java чувствителен к регистру. * Название классов – для всех первая буква должна быть в верхнем регистре. * Если несколько слов используются, чтобы сформировать название класса, первая буква каждого внутреннего слова должна быть в верхнем регистре. * Название методов – все имена методов должны начинаться с буквы нижнего регистра. * Если несколько слов используются, чтобы сформировать имя метода, то первая буква каждого внутреннего слова должна быть в верхнем регистре. * Название файла программы – наименование файла программы должно точно совпадать с именем класса. * При сохранении файла, Вы должны сохранить его, используя имя класса (помните о чувствительности к регистру) и добавить «.java» в конце имени (если имена не совпадают, Ваша программа не будет компилироваться), например, «MyJavaProgram» – это название класса, тогда файл должен быть сохранен как «MyJavaProgram.java». * public static void main(String args[]) – обработка программы начинается с метода main(), который является обязательной частью каждой программы. |
| 2 | Объектно-ориентированное программирование в Джава. Классы в Джава | Созданы классы ChatController (это класс, предназначенный для непосредственной обработки запросов от клиента и возвращения результатов. Задача методов в контроллере - вызов сервисных методов в нужном порядке и возвращение результатов клиенту), Message (это класс, работающий с вводимыми сообщенями пользователей), User (данный класс содержат приватные переменные с данными о польователе, а также этот класс реализует методы для работы с данными о пользователе) |
| 3 | Классы Math и Random. Классы оболочки | Не было реализовано в проекте, выполнено в качестве практической работы |
| 4 | Перечисления и их использование в Джава программах. Наследование в Джава. Абстрактные классы | Перечисление или Java enum — это набор именованных констант, который помогает в определении своих собственных типов данных. В Java это определяет тип класса. Он может иметь конструкторы, методы и переменные экземпляра. В коде проекта также использовался enum (код приведен ниже):  public enum Color { RED, BLACK, YELLOW, CYAN, MAGENTA, BLUE, GREEN, ORANGE, GRAY }  **Наследование** — это механизм в программировании, в том числе и в Java, который позволяет описать новый класс на основе уже существующего. Класс-наследник при этом получает доступ к полям и методам родительского класса. Пример из кода проекта:  public interface UserRepository extends CrudRepository<User, Long> |
| 5 | Создание программ с графическим интерфейсом пользователя на языке Джава | Имеется класс ChatController, в котором по target-у отрисовываются view, то есть каждому адресу сопоставлен графический интерфейс в виде html-страницы. Пример кода:   @GetMapping("/hello") public String showHelloForm() { return "hello"; } @GetMapping("/signup") public String showSignUpForm(User user) { return "add-user"; } @GetMapping("/signin") public String showSignInForm(User user) { return "auth-user"; } |
| 6, 7 | Интерфейсы в Джава. Реализация интерфейсов | Интерфейсы определяют некоторый функционал, не имеющий конкретной реализации, который затем реализуют классы, применяющие эти интерфейсы. И один класс может применить множество интерфейсов. Приведем несколько примернов из нашего проекта:  1. public interface UserRepository extends CrudRepository<User, Long>  { List<User> findByName(String name); Optional<User> findById(Long id); }  2. public interface ChatService { void sendMessage(Message message); User addUser(User user); Iterable<Message> findAllMessages(); Iterable<User> findAllUsers(); } |
| 8 | Рекурсия. Программирование рекурсии в Джава. Решение задач на рекурсию | Не было реализовано в проекте, выполнено в качестве практической работы |
| 9 | Использование полиморфизма при программировании при реализации алгоритмов сортировок и поиска | Сортировка — это процесс упорядочивания списка элементов (организация в определенном порядке) исходного списка элементов, который возможно организован в виде контейнера или храниться в виде коллекции. Не было реализовано в проекте, выполнено в качестве практической работы |
| 10 | Стандартные интерфейсы Джава. Интерфейс Comparator | В языке Java определен ряд стандартных (предопределенных) функциональных интерфейсов. Эти интерфейсы широко используются в стандартных библиотеках языка Java, в частности, в средствах обработки потоков данных StreamAPI. Стандартные функциональные интерфейсы также можно использовать при разработке собственных программ. Далее приведен перечень стандартных функциональных интерфейсов. UnaryOperator<T>, BinaryOperator<T>, Consumer<T> , Supplier<T>, Function<T, R>, Predicate<T>.  Comparator – интерфейс, класс которого реализует функцию сравнения двух объектов. Comparator возвращает int по следующей схеме:   * отрицательный int (первый объект отрицательный, то есть меньше) * положительный int (первый объект положительный, хороший, то есть больший) * ноль = объекты равны   Не было реализовано в проекте, выполнено в качестве практической работы |
| 11 | Работа с датой и временем | Java предоставляет класс Date, который доступен в пакете java.util, этот класс заключает в себе **текущую дату и время**. Он хранит дату в миллисекундах, которые прошли с 1 января 1970 года. Для этой даты есть даже отдельное название — “Unix-время” **SimpleDateFormat**– это конкретный класс для парсинга и форматирования даты в Java. SimpleDateFormat позволяет начать с выбора любых пользовательских шаблонов для форматирования даты и времени. Пример использования их кода проекта:  public void setDate(String date) { date = new SimpleDateFormat("dd-MM-yyyy").format(new Date()); } public String getDate() {return date;} |
| 12 | Создание программ с графическим интерфейсом пользователя на языке Джава. Компоновка объектов с помощью Layout-менеджеров | Имеется класс ChatController, в котором по target-у отрисовываются view, то есть каждому адресу сопоставлен графический интерфейс в виде html-страницы. Пример кода:   @GetMapping("/hello") public String showHelloForm() { return "hello"; } @GetMapping("/signup") public String showSignUpForm(User user) { return "add-user"; } @GetMapping("/signin") public String showSignInForm(User user) { return "auth-user"; }  Коды html-страниц представлены в программе |
| 13, 14 | Обработка строк. Использование регулярных выражений в Джава-приложениях | Регулярное выражение (RegEx) – это шаблон для поиска строки в тексте. Для бработки строки с паролем мы используем регулярные выражения для проверки правильности введенного пароля, чтобы в нем были только латинские буквы в любом из регистров и цифры. Приведем пример кода из проекта:  public boolean isRightPassword (String password) { Pattern pattern = Pattern.compile ("^(?=.\*\\d)(?=.\*[a-z])(?=.\*[A-Z])(?!.\*\\s).\*$"); Matcher matcher = pattern.matcher(password); return matcher.find(); } |
| 15 | Вложенные и внутренние классы. Обработка событий в Джава-программах с графическим интерфейсом пользователя | Имеется класс ChatController, в котором по target-у отрисовываются view, то есть каждому адресу сопоставлен графический интерфейс в виде html-страницы. Когда пользователь вводит свои данные в графический интерфейс и отправляет форму, то отсылается запрос на сервер (backend) и обрабатывается на нем. Пример кода:  @PostMapping("/adduser") public String addUser(User user, Model model, HttpServletRequest request) { try { User savedUser = chatService.addUser(user); request.getSession().setAttribute("userName", savedUser.getName()); request.getSession().setAttribute("color", savedUser.getColor()); request.getSession().setAttribute("password", savedUser.<String>getPassword()); model.addAttribute("users", chatService.findAllUsers()); } catch (Exception e) { model.addAttribute("error", "That user name already exist, please authorized!"); return "auth-user"; } return "redirect:/"; } |
| 16 | Обработка событий мыши и клавиатуры в программах на Джава с графическим интерфейсом пользователя | Во время регистрации пользователя он нажимает на кнопку «Зарегистрироваться» и отправляется POST-запрос на сервер. Получается, на этой кнопке стоит функция отслеживания нажатия мыши (встроенная в Html). При отправке сообщения пользователь может как нажать на кнопку «Отправить» мышью, так и нажав клавишу “Enter” на клавиатуре, после чего происходит отправка запроса на сервер. Пример кода для событий мыши:  <input type="submit" class="btn btn-primary" value="Add User">  Пример кода для обработки события клавиатуры:  <input type="submit" class="chat-form\_\_submit" value="Send"> |
| 17 | Разработка интерактивных программ на языке Джава с использованием паттерна MVC | MVC — набор архитектурных идей и принципов для построения сложных систем с пользовательским интерфейсом.  Model. Первая компонента/модуль — так называемая модель. Она содержит всю бизнес-логику приложения. View. Вторая часть системы — вид. Данный модуль отвечает за отображение данных пользователю. Все, что видит пользователь, генерируется видом. Controller. Третьим звеном данной цепи является контроллер. В нем хранится код, который отвечает за обработку действий пользователя (любое действие пользователя в системе обрабатывается в контроллере). Создадим главную страницу нашего проекта - index.html, а также страницы add-user.html, auth-user.html и hello.html. Создадим класс ChatController. Аннотация @Controller - говорит о том, что данный класс является контроллером. Контроллеры в Spring обрабатывают HTTP запросы на определенный адреса. Также имеем файл pom.xml - дескриптор развертывания, который позволяет быстро и просто импортировать библиотеки из разных фреймвроков в проект, а также, в котором мы конфигурируем сборку. Проект собирается с помощью Maven, pom.xml - конфигурационный файл этой системы сборки. |
| 18, 19 | Исключения и работа с ними в Джава. Создание пользовательских исключений | Исключение в Java - это объект, который описывает исключительное состояние, возникшее в каком-либо участке программного кода. Когда возникает исключительное состояние, создается объект класса Exception. Этот объект пересылается в метод, обрабатывающий данный тип исключительной ситуации. Исключения могут возбуждаться и для того, чтобы сообщить о некоторых нештатных ситуациях. Приведем пример из кода проекта:  try { User savedUser = chatService.addUser(user); request.getSession().setAttribute("userName", savedUser.getName()); request.getSession().setAttribute("color", savedUser.getColor()); request.getSession().setAttribute("password", savedUser.getPassword()); model.addAttribute("users", chatService.findAllUsers()); } catch (Exception e) { model.addAttribute("error", "That user name already exist, please authorized!"); return "auth-user"; } return "redirect:/"; |
| 20 | Работа с дженериками | **Дженерики** (обобщения) — это особые средства языка Java для реализации обобщённого программирования: особого подхода к описанию данных и алгоритмов, позволяющего работать с различными типами данных без изменения их описания. <T> в сигнатуре метода подразумевает, что метод будет иметь дело с универсальным типом T. Это необходимо, даже если метод возвращает void. Приведем пример кода нашего проекта:  public Iterable<Message> findAllMessages() { return messageRepository.findAll(); } |
| 21 | Стирание типов в Джава | Cуть стирания типов заключается в том, что внутри класса не хранится никакой информации о его типе-параметре. Эта информация доступна только на этапе компиляции и стирается (становится недоступной) в runtime. Не было реализовано в проекте, выполнено в качестве практической работы. |
| 22 | Абстрактные типы данных. Стек | Абстрактный тип данных – это тип данных, который разработчик создает для работы с определенной сущьностью. В классе User имеются свои поля и методы, которые определены в данном классе, для работы с его полями.  Структура данных — программная единица, позволяющая хранить и обрабатывать множество однотипных и/или логически связанных данных в вычислительной технике. Стеком называется линейная динамическая структура данных, добавление и исключение элементов в которую и производится с одного конца, называемого вершиной стека. Стек работает по принципу LIFO (Last-In, First-Out). |
| 23 | Абстрактные типы данных. Очередь | Абстрактный тип данных – это тип данных, который разработчик создает для работы с определенной сущьностью. В классе User имеются свои поля и методы, которые определены в данном классе, для работы с его полями.  Очереди представляют структуру данных, работающую по принципу FIFO (first in - first out). То есть чем раньше был добавлен элемент в коллекцию, тем раньше он из нее удаляется. |
| 24 | Паттерны проектирования. Порождающие паттерны: абстрактная фабрика, фабричный метод | Паттерны проектирования (или шаблоны) – это описание проблем, которые встречаются при написании объектно-ориентированного кода, а так же примеры решения этих проблем. Шаблон проектирования фабричный метод решает проблему создания различных групп продуктов, каждая из которых обладает некоторой спецификой. Абстрактная фабрика, как и все фабричные паттерны, помогает нам правильно организовать создание новых объектов. С ее помощью мы управляем “выпуском” различных семейств взаимосвязанных объектов.  Шаблоны проектирования Джава делятся на три категории: порождающие, структурные и поведенческие шаблоны проектирования. Так же есть еще шаблоны проектировани, например MVC – model-view-controller |
| 25 | Классы Pattern, Matcher и PatternSyntaxException | Первый метод принимает один аргумент – строковый литерал нашего регулярного выражения. Matcher сопоставляет символ шаблона с каждым символом текста, начиная с нулевой позиции. Приведем пример кода из проекта:  public boolean isRightPassword (String password) { Pattern pattern = Pattern.compile("^(?=.\*\\d)(?=.\*[a-z])(?=.\*[A-Z])(?!.\*\\s).\*$"); Matcher matcher = pattern.matcher(password); return matcher.find(); } |
| 26, 27, 28, 29 | JavaCollectionFramework: нелинейные структуры данных | Интерфейс Collection — это то место, откуда берут начало все коллекции (в него входят List, Set, Queue). Collection — это идея, это представление о том, как должны себя вести все коллекции. Java Collection Framework — иерархия интерфейсов и их реализаций, которая является частью JDK и позволяет разработчику пользоваться большим количесвом структур данных из «коробки». Списки (List) выражают то, что у нас есть коллекция элементов, которые расположены в некоторой последовательности друг за другом. Каждый элемент имеет индекс (как в массиве). Как правило, список позволяет иметь элементы с одинаковым значением. Пример List из нашего кода:  List<User> findByName(String name);  Очередь (Queue) — это структура, знакомая нам из жизни. Очереди в магазины, к врачам. Кто первее пришёл (First In), тот первее и выйдет из очереди (First Out).  Set — переводится как "набор". От очереди и списка Set отличается большей абстракцией над хранением элементов. Set — как мешок с предметами, где неизвестно, как лежат предметы и в каком порядке они легли. |
| 30 | Разработка программы по UML | Нами была расписана архитектура проекта в виде UML-диаграммы, она представлена на рисунке ниже. |
| 31 | Тестирование | Тестирование изображено на рисунке ниже. |
| 32 | Сериализация | Не было реализовано в проекте, выполнено в качестве практической работы. |



**Рисунок 1 - UML-диаграмма проекта**



**Рисунок 2 - Тестирование**

Задание 3. Классы Math и Random. Классы оболочки

import java.util.Random;  
  
public class Main {  
 public static void main(String[] args) {  
 int randomInt = new Random().ints(1, 1, 11).findFirst().getAsInt();  
 double res = Math.*pow*(randomInt, 4);  
 String strRes = Double.*toString*(res);  
 System.*out*.println(res);  
 }  
}

Задание 8. Рекурсия. Программирование рекурсии в Джава. Решение задач на рекурсию

import java.util.Scanner;  
  
public class Main {  
 public static void main(String[] args) {  
 System.*out*.println("Введите n - номер числа Фибоначчи, которое необходимо получить");  
 Scanner in = new Scanner(System.*in*);  
 int n = in.nextInt();  
 int numFibonacci = *getFibonacciValue*(n);  
 System.*out*.println(numFibonacci);  
 }  
 public static int getFibonacciValue(int n) {  
 if (n == 0) {  
 return 0;  
 } else if (n == 1) {  
 return 1;  
 } else {  
 return *getFibonacciValue*(n - 1) + *getFibonacciValue*(n - 2);  
 }  
 }  
}

Задание 9 и 10. Интерфейс Comparator

import java.io.\*;  
import java.lang.\*;  
import java.util.\*;  
class Student {

String name, address;  
 public Student(int rollno, String name, String address)  
 {  
 this.rollno = rollno;  
 this.name = name;  
 this.address = address;  
 }  
 public String toString()  
 {  
 return this.rollno + " " + this.name + " "  
 + this.address;  
 }  
}  
class Sortbyname implements Comparator<Student> {  
 public int compare(Student a, Student b)  
 {  
  
 return a.name.compareTo(b.name);  
 }  
}  
class GFG {  
 public static void main(String[] args)  
 {  
 ArrayList<Student> ar = new ArrayList<Student>();  
 ar.add(new Student(111, "Kate", "london"));  
 ar.add(new Student(131, "Grege", "nyc"));  
 System.*out*.println("Unsorted");  
 for (int i = 0; i < ar.size(); i++)  
 System.*out*.println(ar.get(i));  
 Collections.*sort*(ar, new Sortbyname());  
 System.*out*.println("\nSorted by name");  
 for (int i = 0; i < ar.size(); i++)  
 System.*out*.println(ar.get(i));  
 }  
}

Задание 21. Стирание типов в Джава

import java.util.ArrayList;  
import java.util.List;  
public class Main {  
 private class Cat {  
 }  
 public static void main(String[] args) {  
 List<String> strings = new ArrayList<>();  
 List<Integer> numbers = new ArrayList<>();  
 List<Cat> cats = new ArrayList<>();  
 System.*out*.println(strings.getClass() == numbers.getClass());  
 System.*out*.println(numbers.getClass() == cats.getClass());  
 }  
}

Задание 22. Стэк

import java.util.Stack;  
class Main {  
 public static void main(String[] args) {  
 Stack<String> name= new Stack<>();  
 name.push("Kate");  
 name.push("Nastya");  
 name.push("Slava");  
 System.*out*.println("Stack: " + name);  
 name.pop();  
 System.*out*.println("Stack after pop: " + name);  
 name.pop();  
 System.*out*.println("Stack after pop: " + name);  
 name.pop();  
 System.*out*.println("Stack after pop: " + name);  
 }  
}

Задание 23 и 29. Очередь

import java.util.Queue;  
import java.util.LinkedList;  
class Main {  
 public static void main(String[] args) {  
 Queue<String> name = new LinkedList<>();  
 name.offer("Nastya");  
 name.offer("Kate");  
 name.offer("Slava");  
 System.*out*.println("Queue: " + name);  
 String removedName = name.poll();  
 System.*out*.println("Removed Element: " + removedName);  
 System.*out*.println("Queue after deletion: " + name);  
 }  
}

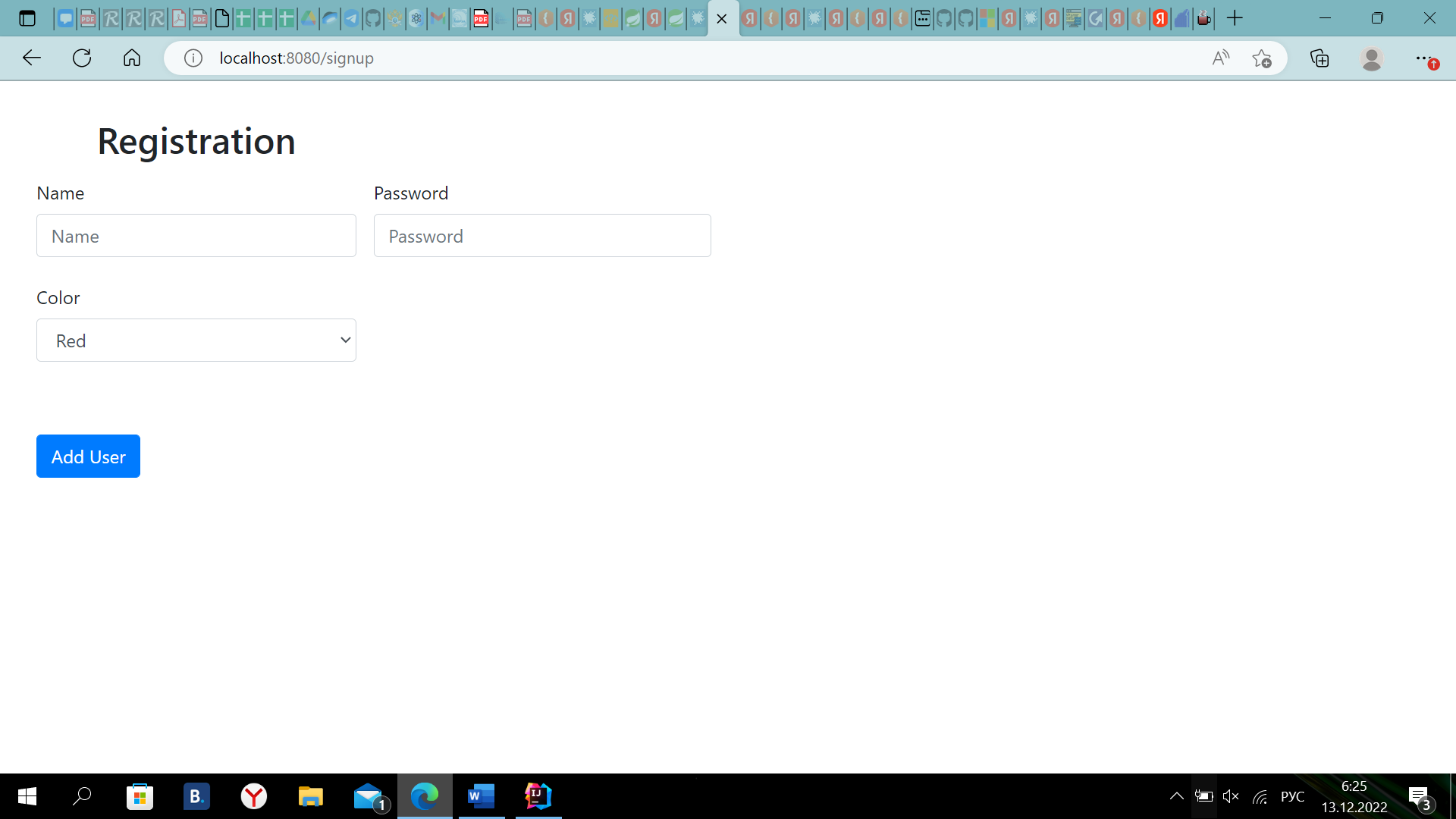
Задание 32. Сериализация.

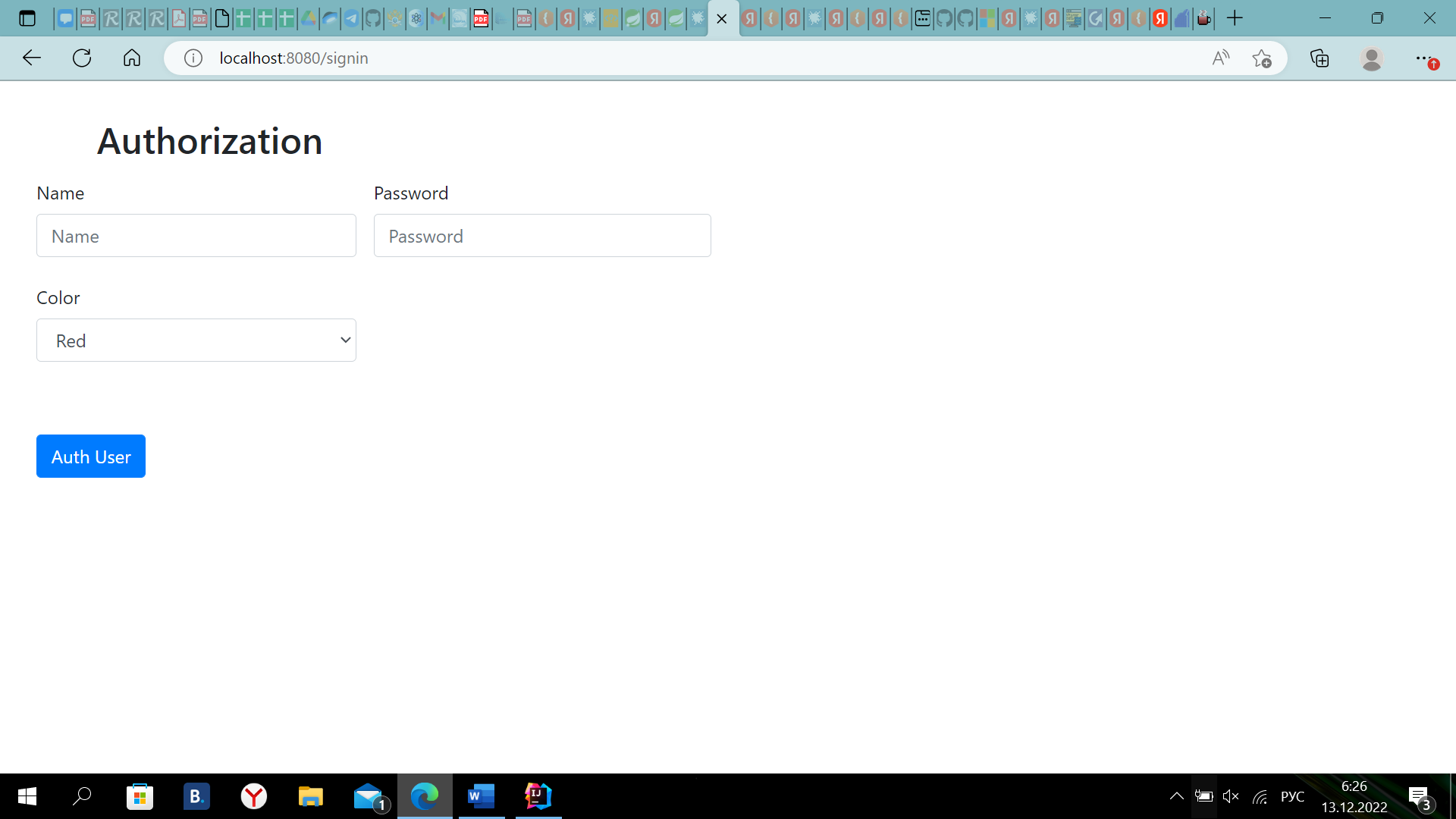
import java.io.FileOutputStream;  
import java.io.IOException;  
import java.io.ObjectOutputStream;  
import java.io.Serializable;  
  
class parent implements Serializable {  
 int parentVersion = 10;  
}  
  
class contain implements Serializable{  
 int containVersion = 11;  
}  
public class SerialTest extends parent implements Serializable {  
 int version = 66;  
 contain con = new contain();  
  
 public int getVersion() {  
 return version;  
 }  
 public static void main(String args[]) throws IOException {  
 FileOutputStream fos = new FileOutputStream("temp.out");  
 ObjectOutputStream oos = new ObjectOutputStream(fos);  
 SerialTest st = new SerialTest();  
 oos.writeObject(st);  
 oos.flush();  
 oos.close();  
 }  
}

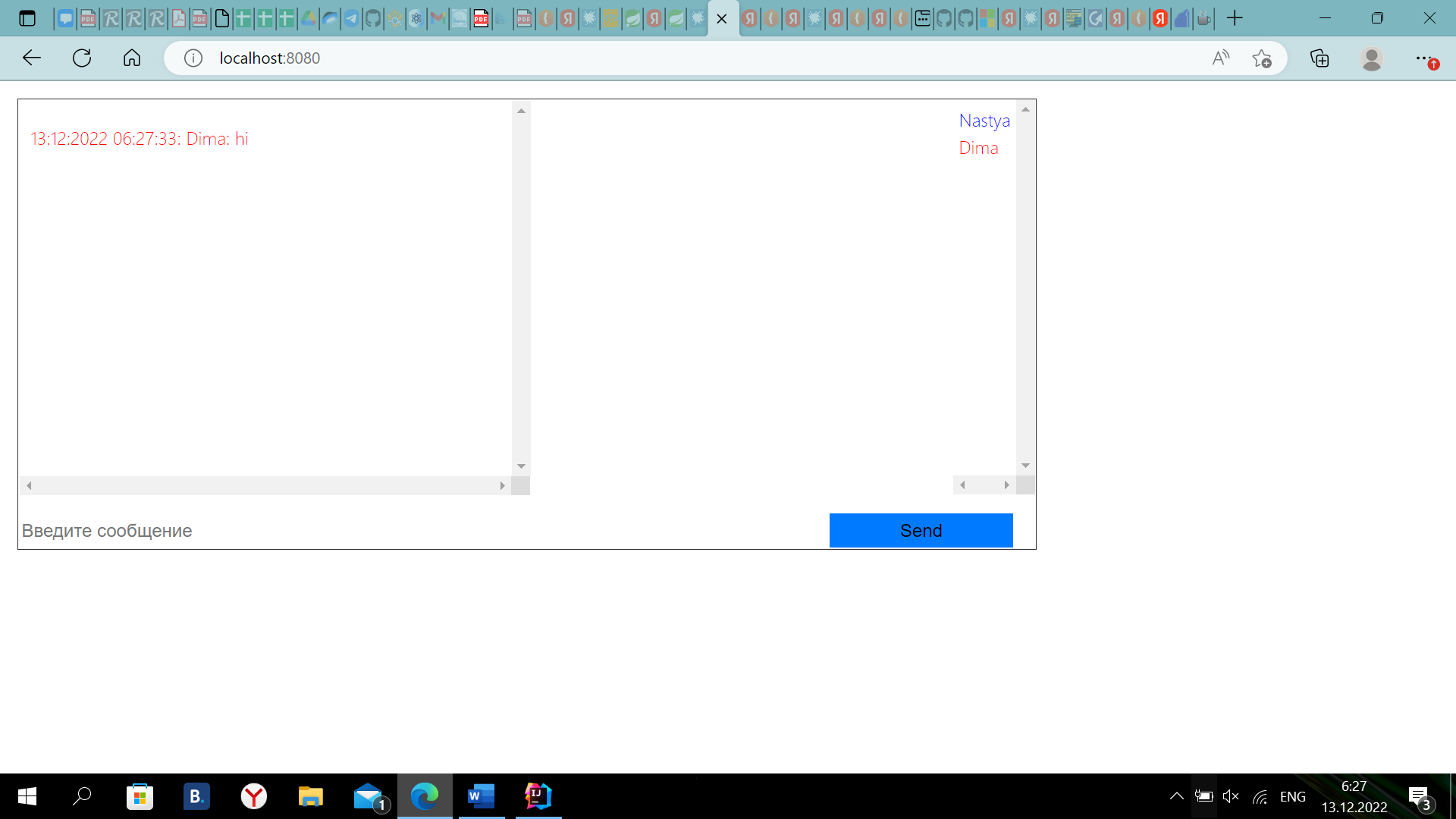
## Код программы

https://github.com/Nastuysha/webchat/tree/master

## Вывод программы







## Вывод

В результате выполнения проекта получены навыки работы в IntelliJ IDEA, создан Web-chat на языке Джава, изучены паттерны и разработка программы по UML, а также проведено автотестирование проекта.