МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА ИНФОРМАЦИОННО-СЕТЕВЫХ ТЕХНОЛОГИЙ

КУРСОВАЯ РАБОТА (ПРОЕКТ) ЗАЩИЩЕНА С ОЦЕНКОЙ

РУКОВОДИТЕЛЬ

ст. препод, канд. техн. наук П. А. Степанов

должность, уч. степень, звание подпись, дата инициалы, фамилия

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОЙ РАБОТЕ

по дисциплине: УПРАВЛЕНИЕ КАЧЕСТВОМ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. № 4831 К. А. Корнющенков

подпись, дата инициалы, фамилия

Санкт-Петербург 2021

**Оглавление**

[ВВЕДЕНИЕ 2](#_Toc72412800)

[Цель работы 3](#_Toc72412801)

[Задача 3](#_Toc72412802)

[Реализация проекта с системой контроля версий и непрерывной интеграцией 4](#_Toc72412803)

[Репозиторий GitHub 4](#_Toc72412804)

[Непрерывная интеграция 6](#_Toc72412805)

[ЗАКЛЮЧЕНИЕ 7](#_Toc72412806)

[ПРИЛОЖЕИЕ A 8](#_Toc72412807)

[ПРИЛОЖЕНИЕ B 19](#_Toc72412808)

[ПРИЛОЖЕНИЕ C 20](#_Toc72412809)

# ВВЕДЕНИЕ

Система управления версиями (от англ. Version Control System, VCS или Revision Control System) — программное обеспечение для облегчения работы с изменяющейся информацией. Система управления версиями позволяет хранить несколько версий одного и того же документа, при необходимости возвращаться к более ранним версиям, определять, кто и когда сделал то или иное изменение, и многое другое.

Такие системы наиболее широко используются при разработке программного обеспечения для хранения исходных кодов разрабатываемой программы.

Однако они могут с успехом применяться и в других областях, в которых ведётся работа с большим количеством непрерывно изменяющихся электронных документов. В частности, системы управления версиями применяются в САПР, обычно в составе систем управления данными об изделии (PDM). Управление версиями используется в инструментах конфигурационного управления (Software Configuration Management Tools).

Непрерывная интеграция (CI, англ. Continuous Integration)— практика разработки программного обеспечения, которая заключается в постоянном слиянии рабочих копий в общую основную ветвь разработки (до нескольких раз в день) и выполнении частых автоматизированных сборок проекта для скорейшего выявления потенциальных дефектов и решения интеграционных проблем. В обычном проекте, где над разными частями системы разработчики трудятся независимо, стадия интеграции является заключительной. Она может непредсказуемо задержать окончание работ.

Переход к непрерывной интеграции позволяет снизить трудоёмкость интеграции и сделать её более предсказуемой за счёт наиболее раннего обнаружения и устранения ошибок и противоречий, но основным преимуществом является сокращение стоимости исправления дефекта, за счёт раннего его выявления.

# Цель работы

В рамках выполнения курсовой работы необходимо выполнить следующие работы

* Взять свою лабораторную работу по этому курсу и создать для нее репозиторий. Выложить на github.
* Разработать план выпуска новой версии (только содержательную часть)
* Оформить работы по выпуску новой версии в виде списка issues
* Создать скелет функционала новой программы (классы и экранные формы).
* Создать тесты на новый функционал. Отделить тесты на реализованный функционал от нереализованных.
* Создать план по тестированию (артефакт согласно PMBOK)
* Настроить систему непрерывной интеграции. Сделать так, чтобы тесты на реализованный функционал выполнялись в ответ на интеграцию.
* Создать отчет по курсовому проектированию.

# Задача



# Реализация проекта с системой контроля версий и непрерывной интеграцией

### Репозиторий GitHub

Созданная в рамках лабораторной работы программа была загружена в репозиторий Github с главной веткой main, по адресу

<https://github.com/Kiryakor/QA>

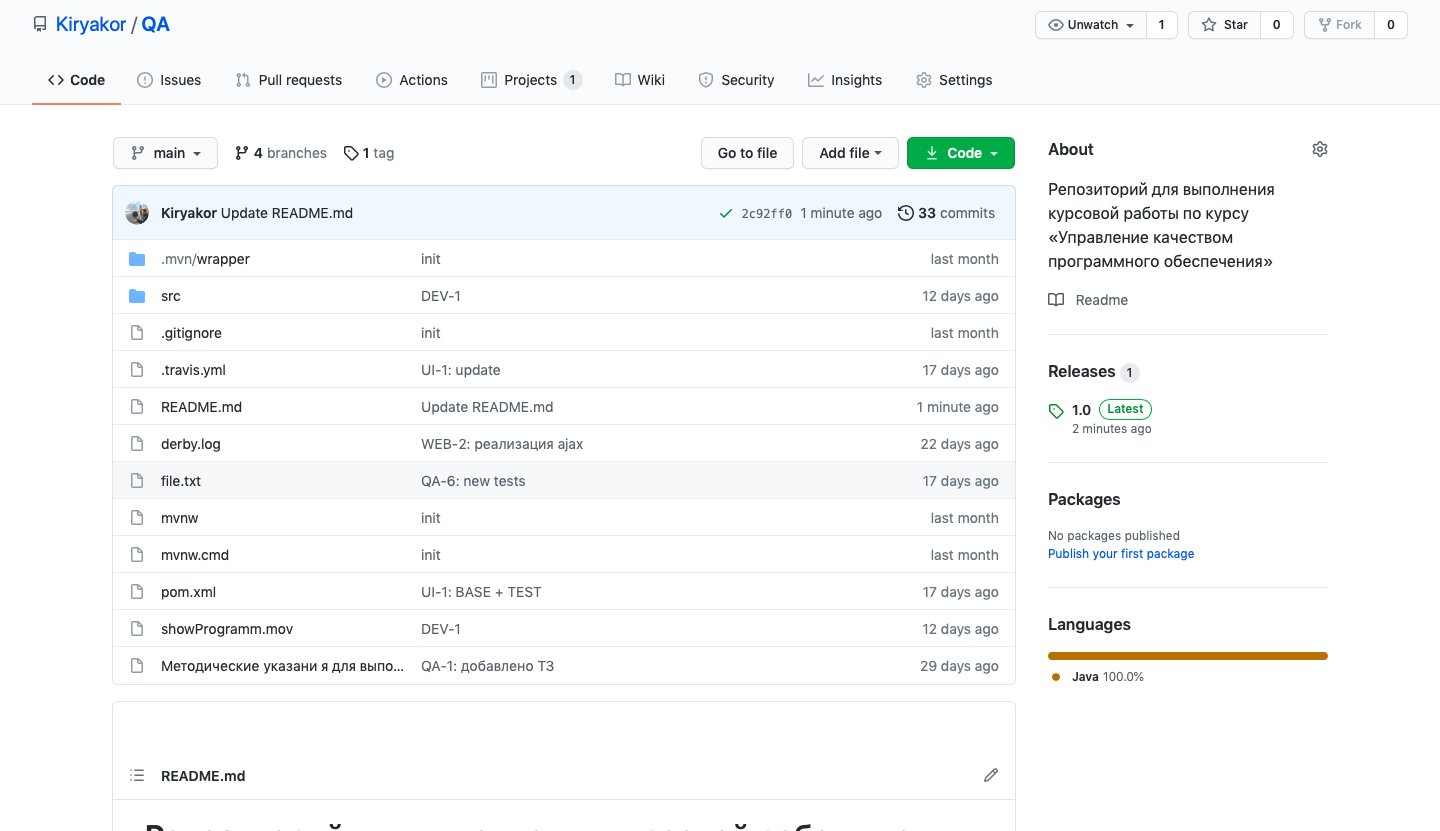


Рисунок 1: Вид репозитория

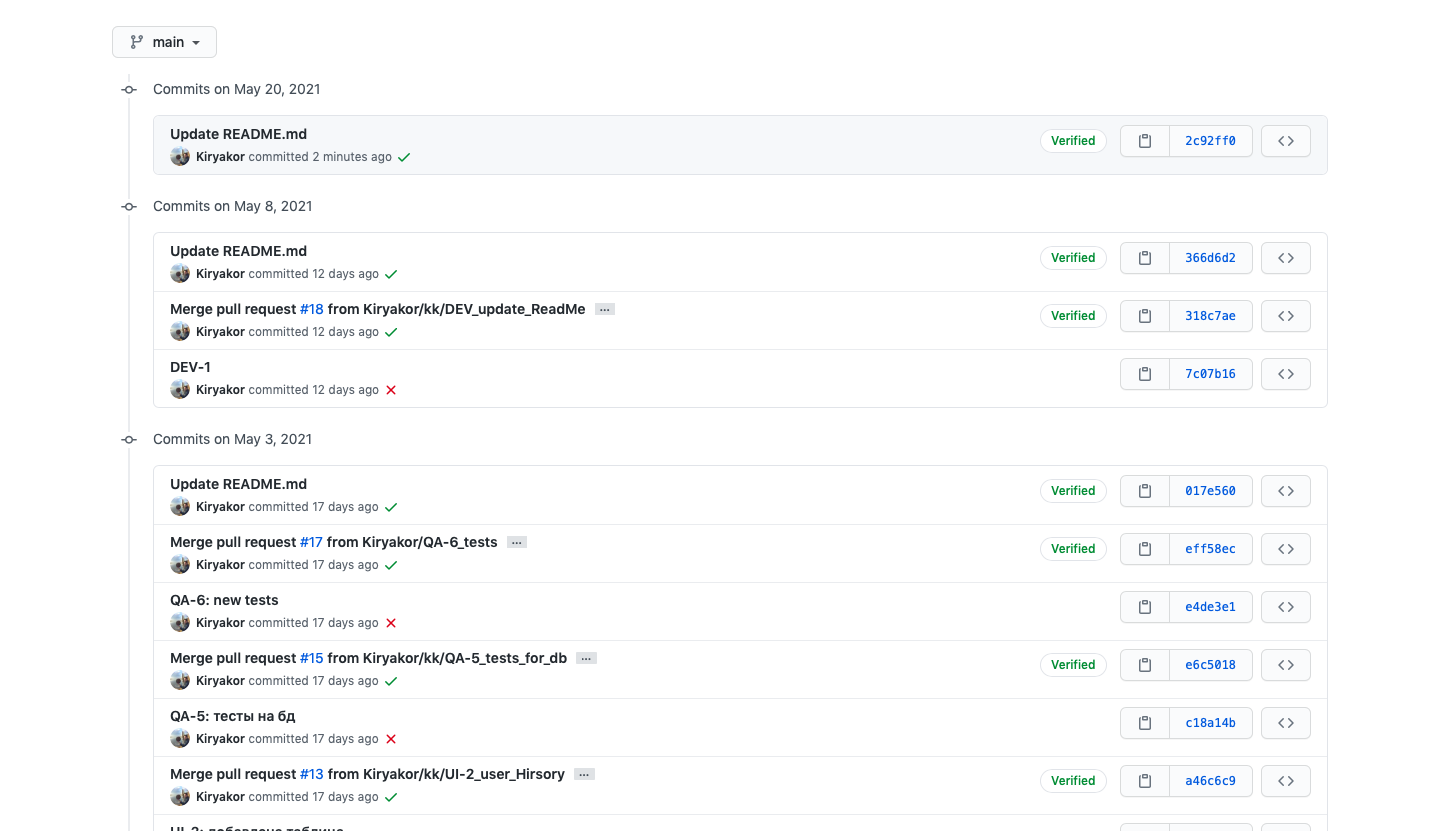


Рисунок 2: Коммиты в репозитории

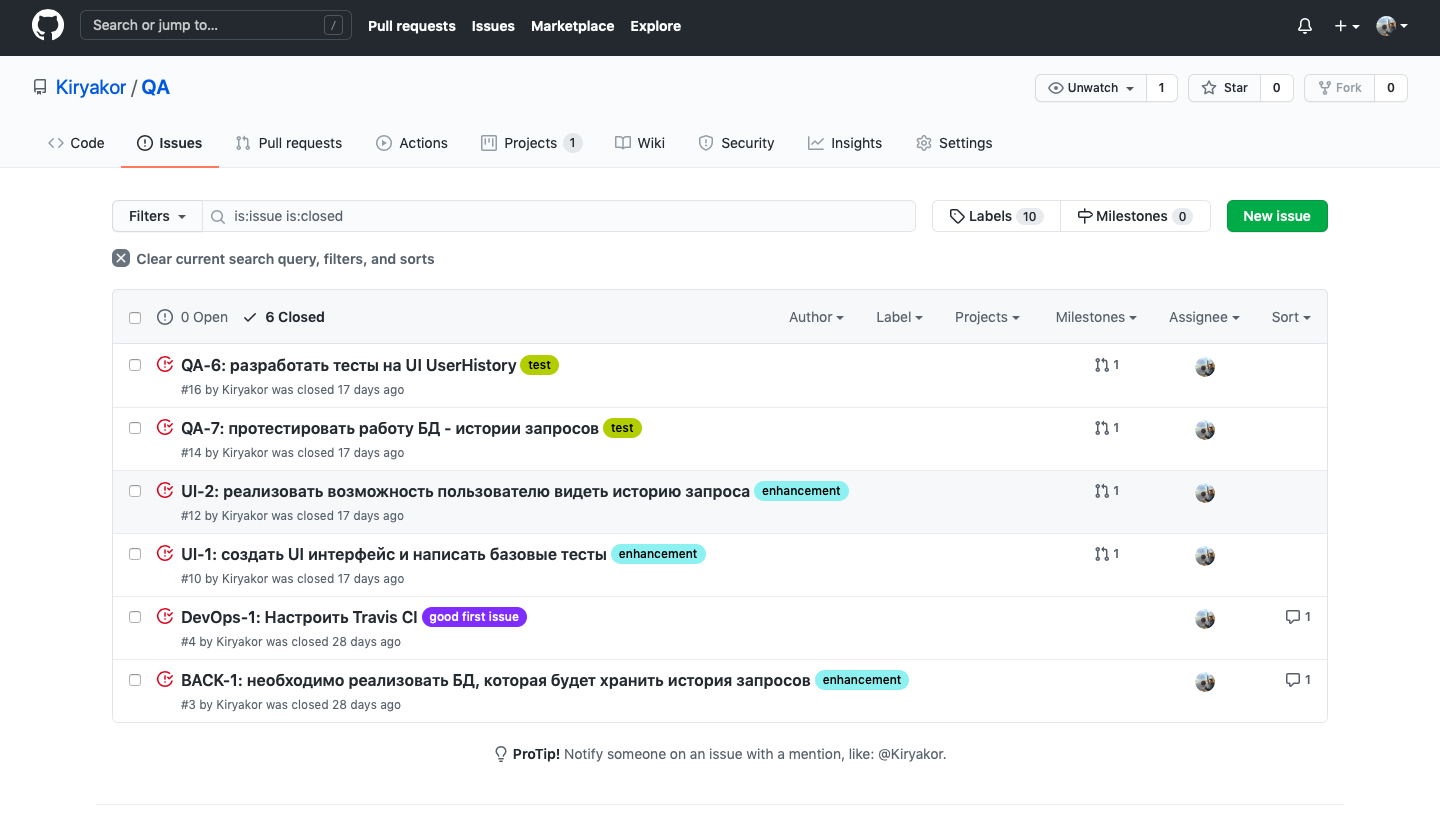


Рисунок 3: Скриншот созданных и закрытых issues

### Непрерывная интеграция

Непрерывная интеграция для курсовой работы была настроена с помощью сервиса GitHub – Github Actions.

Код запуска тестов находится в приложении B

Интеграция осуществляется при помощи yml файла, который запускает сборку проекта при коммитах и пул-реквестах в мастер ветку.

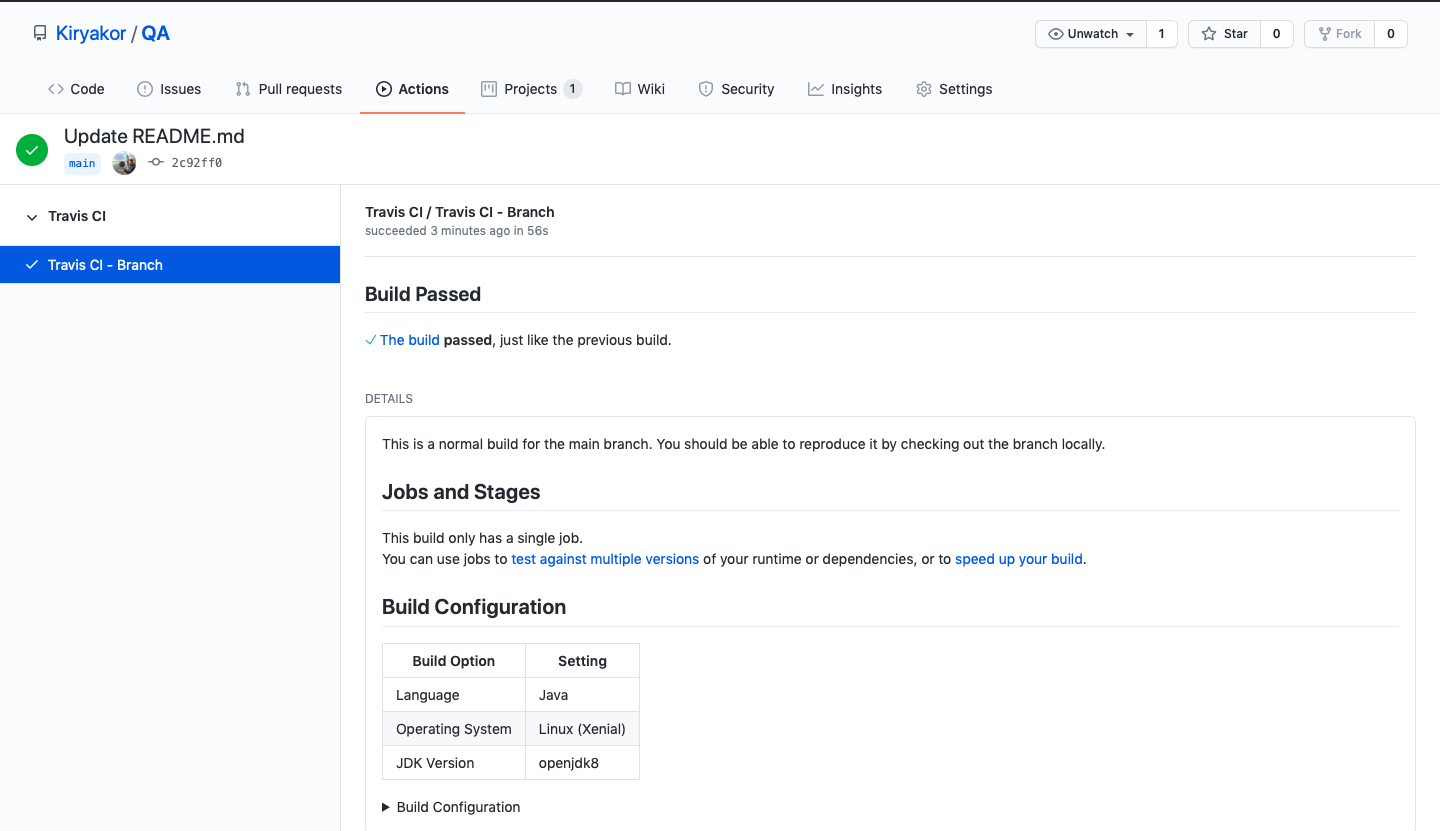


Рисунок 4: Скриншот пройденного теста в системе GitHub Actions

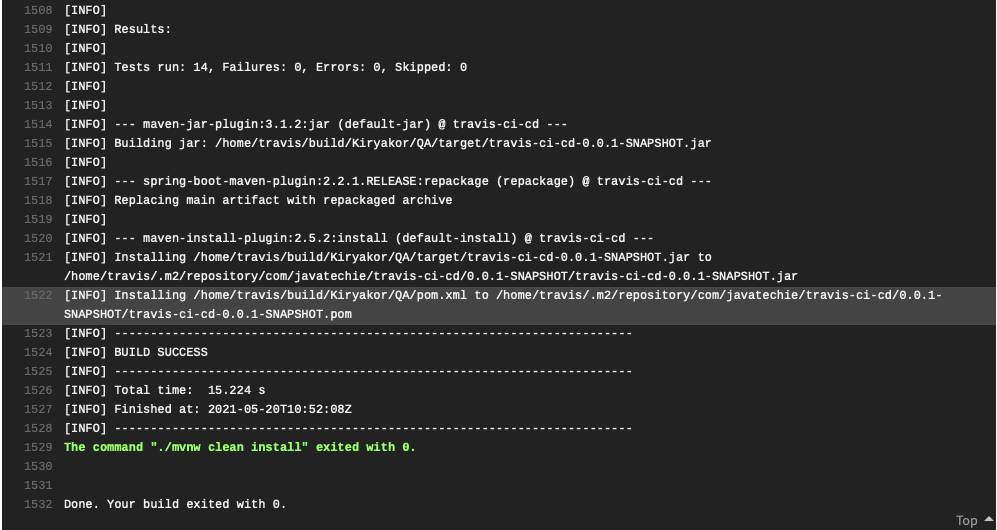


Рисунок 5: Скриншот логов прохождения тестов GitHub Actions

# ЗАКЛЮЧЕНИЕ

В рамках данной курсовой работы лабораторная работа была загружена в репозиторий, для неё был создан план выпуска новых версий на репозитории GitHub, выпущены новые версии и настроена непрерывная интеграция с использованием GitHub Actions. Написан план по тестированию и вставлен в приложении «C».

# ПРИЛОЖЕИЕ A

**Листинг программы**

SimpleGUI

package com.javatechie.travis.api;

import com.javatechie.travis.api.parser.MyParser;

import com.javatechie.travis.api.userHistory.UserHistory;

import com.javatechie.travis.api.userHistory.UserHistoryServiceImp;

import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

public class SimpleGUI extends JFrame {

private JButton button = new JButton("Press");

private JTextField input = new JTextField("", 5);

private JLabel label = new JLabel("");

private JTable table;

private DefaultTableModel tableModel;

private Object[] columnsHeader = new String[] {"Запрос", "Ответ"};

private Object[][] userHistoryArray;

MyParser myParser = new MyParser();

UserHistoryServiceImp userHistoryService = new UserHistoryServiceImp();

public SimpleGUI() {

super("QA");

this.setBounds(100,100,250,600);

this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

Container container = this.getContentPane();

container.setLayout(new GridLayout(4,1));

container.add(input);

label.setName("anwerLabel");

container.add(label);

Object[] columnsHeader = new String[] {"Наименование", "Ед.измерения"};

tableModel = new DefaultTableModel();

tableModel.setColumnIdentifiers(columnsHeader);

table = new JTable(tableModel);

table.setName("table");

prepareData();

container.add(table);

button.setName("actionButton");

button.addActionListener(new ButtonEventListener());

container.add(button);

}

class ButtonEventListener implements ActionListener {

public void actionPerformed(ActionEvent e) {

String message = "";

userHistoryService.add(input.getText());

message = myParser.action(input.getText());

label.setText(message);

Object[] columnsHeader = new String[] { input.getText(), message};

input.setText("");

tableModel.addRow(columnsHeader);

table.updateUI();

}

}

public void prepareData() {

try {

Iterable<UserHistory> data = userHistoryService.listAll();

for (UserHistory t : data) {

Object[] columnsHeader = new String[]{t.getAnswer(), t.getRequest()};

tableModel.addRow(columnsHeader);

}

} catch (Exception e) {

System.out.println(e.getLocalizedMessage());

}

}

public static void main(String[] args) {

SimpleGUI app = new SimpleGUI();

app.setVisible(true);

}

}

**MyParser**

package com.javatechie.travis.api.parser;

import java.util.LinkedList;

public class MyParser {

public static String action(String data){

if (checkData(data)){

Pair<LinkedList<Integer>, LinkedList<String>> parse = parseData(data);

parse = hardAction(parse.t,parse.u);

return simpleAction(parse.t,parse.u);

}else {

return "Не вырные входные данные";

}

}

private static boolean isNumeric(char str) {

try {

Double.parseDouble(String.valueOf(str));

return true;

} catch(NumberFormatException e){

return false;

}

}

private static Boolean checkData(String data){

for (int i=0;i<data.length();i++){

if ((i ==0 || i == data.length() - 1) && !isNumeric(data.charAt(i))) {

return false;

}

if (isNumeric(data.charAt(i)) | String.valueOf(data.charAt(i)).equals("+")

| String.valueOf(data.charAt(i)).equals("\*") | String.valueOf(data.charAt(i)).equals("/") | String.valueOf(data.charAt(i)).equals("-")){

continue;

} else {

return false;

}

}

return true;

}

private static Pair<LinkedList<Integer>, LinkedList<String>> parseData(String data){

LinkedList<Integer> number = new LinkedList<Integer>();

LinkedList<String> action = new LinkedList<String>();

String timeNumber = "";

for (int i=0;i<data.length();i++){

if (!isNumeric(data.charAt(i))){

action.add(String.valueOf(data.charAt(i)));

if (timeNumber != "") {

number.add(Integer.valueOf(timeNumber));

timeNumber = "";

}

} else{

timeNumber += String.valueOf(data.charAt(i));

}

}

if (timeNumber != "") {

number.add(Integer.valueOf(timeNumber));

timeNumber = "";

}

return new Pair(number,action);

}

private static Pair<LinkedList<Integer>, LinkedList<String>> hardAction(LinkedList<Integer> number, LinkedList<String> action){

Integer i = 0;

Boolean check = true;

while (check){

if (action.get(i).equals("\*")) {

number.set(i,number.get(i)\*number.get(i+1));

number.remove(i+1);

action.remove(i+1-1);

i = 0;

}else if (action.get(i).equals("/")) {

number.set(i,number.get(i)/number.get(i+1));

number.remove(i+1);

action.remove(i+1-1);

i = 0;

}

i += 1;

if (action.size() <= i){

break;

}

}

return new Pair(number,action);

}

private static String simpleAction(LinkedList<Integer> number, LinkedList<String> action){

Integer i = 0;

while (!action.isEmpty()){

if (action.get(i).equals("+")) {

number.set(i,number.get(i)+number.get(i+1));

number.remove(i+1);

action.remove(i+1-1);

i = 0;

}else if (action.get(i).equals("-")) {

number.set(i,number.get(i)-number.get(i+1));

number.remove(i+1);

action.remove(i+1-1);

i = 0;

}

}

return String.valueOf(number.get(0));

}

}

Pair

package com.javatechie.travis.api.parser;

public class Pair<T,U> {

public final T t;

public final U u;

public Pair(T t, U u) {

this.t= t;

this.u= u;

}

}

**UserHistory**

package com.javatechie.travis.api.userHistory;

public class UserHistory {

public UserHistory(String answer, String request) {

this.answer = answer;

this.request = request;

}

public UserHistory(){}

private String answer;

private String request;

public String getAnswer() {

return answer;

}

public String getRequest() {

return request;

}

}

**UserHistoryRepository**

package com.javatechie.travis.api.userHistory;

import java.io.\*;

import java.nio.charset.StandardCharsets;

import java.nio.file.Files;

import java.nio.file.Paths;

import java.util.LinkedList;

import java.util.List;

public class UserHistoryRepository {

public String path = "file.txt";

public Iterable<UserHistory> findAll() {

LinkedList<UserHistory> userHistoriesList = new LinkedList<UserHistory>();

List<String> data;

try {

data = readFile();

for (String std : data) {

userHistoriesList.add(new UserHistory(std.split("\t")[0],std.split("\t")[1]));

}

} catch (IOException e) {

}

return userHistoriesList;

}

public UserHistory save(UserHistory userHistory){

try {

write(userHistory);

} catch (IOException e) {

}

return userHistory;

}

private void write(UserHistory userHistory) throws IOException {

try {

List<String> readData = readFile();

readData.add(userHistory.getAnswer() + "\t" + userHistory.getRequest());

BufferedWriter writer = new BufferedWriter(new FileWriter(path));

for(String str : readData) {

writer.write(str);

writer.write(System.getProperty("line.separator"));

}

writer.close();

} catch (IOException e) {

}

}

private List<String> readFile() throws IOException {

return Files.readAllLines(Paths.get(path), StandardCharsets.UTF\_8);

}

}

**UserHistoryService**

package com.javatechie.travis.api.userHistory;

public interface UserHistoryService {

Iterable<UserHistory> listAll();

UserHistory add(String answer);

}

**UserHistoryServiceImp**

package com.javatechie.travis.api.userHistory;

import com.javatechie.travis.api.parser.MyParser;

public class UserHistoryServiceImp implements UserHistoryService {

UserHistoryRepository userHistoryRepository = new UserHistoryRepository();

MyParser myParser = new MyParser();

@Override

public Iterable<UserHistory> listAll() {

return userHistoryRepository.findAll();

}

@Override

public UserHistory add(String answer) {

String request = myParser.action(answer);

return userHistoryRepository.save(new UserHistory(answer, request));

}

}

**TravisCINewBaseTests**

package com.javatechie.travis.api.newTests;

import com.javatechie.travis.api.SimpleGUI;

import org.assertj.swing.edt.GuiActionRunner;

import org.assertj.swing.fixture.FrameFixture;

import org.junit.Assert;

class TravisCINewBaseTests {

@org.junit.jupiter.api.Test

void simpleTest() {

SimpleGUI frame = GuiActionRunner.execute(() -> new SimpleGUI());

FrameFixture window = new FrameFixture(frame);

window.show();

window.textBox().enterText("5+5+1-2");

window.button("actionButton").click();

Assert.assertEquals("9", window.label().text());

window.cleanUp();

}

@org.junit.jupiter.api.Test

void simpleTest2() {

SimpleGUI frame = GuiActionRunner.execute(() -> new SimpleGUI());

FrameFixture window = new FrameFixture(frame);

window.show();

window.textBox().enterText("5+1-2");

window.button("actionButton").click();

Assert.assertEquals("4", window.label().text());

window.cleanUp();

}

@org.junit.jupiter.api.Test

void simpleTest3() {

SimpleGUI frame = GuiActionRunner.execute(() -> new SimpleGUI());

FrameFixture window = new FrameFixture(frame);

window.show();

window.textBox().enterText("+1-2");

window.button("actionButton").click();

Assert.assertEquals("Не вырные входные данные", window.label().text());

window.cleanUp();

}

@org.junit.jupiter.api.Test

void simpleTest4() {

SimpleGUI frame = GuiActionRunner.execute(() -> new SimpleGUI());

FrameFixture window = new FrameFixture(frame);

window.show();

window.textBox().enterText("1^2");

window.button("actionButton").click();

Assert.assertEquals("Не вырные входные данные", window.label().text());

window.cleanUp();

}

}

**TravisCINewUserHistoryRepositoryTests**

package com.javatechie.travis.api.newTests;

import com.javatechie.travis.api.userHistory.UserHistory;

import com.javatechie.travis.api.userHistory.UserHistoryRepository;

import org.junit.Assert;

import java.io.BufferedWriter;

import java.io.FileWriter;

import java.io.IOException;

import java.nio.charset.StandardCharsets;

import java.nio.file.Files;

import java.nio.file.Paths;

import java.util.List;

public class TravisCINewUserHistoryRepositoryTests {

@org.junit.jupiter.api.Test

void action\_1() {

UserHistoryRepository userHistoryRepository = new UserHistoryRepository();

userHistoryRepository.save(new UserHistory("7+6", "13"));

try {

List<String> readFiles = readFile();

String anser = "7+6" + "\t" + "13";

String wrilteLase = readFiles.get(readFiles.size() - 1);

readFiles.remove(readFiles.size() - 1);

write(readFiles);

Assert.assertEquals(wrilteLase, anser);

} catch (IOException e){}

}

@org.junit.jupiter.api.Test

void action\_2() {

UserHistoryRepository userHistoryRepository = new UserHistoryRepository();

userHistoryRepository.save(new UserHistory("5+6", "11"));

try {

List<String> readFiles = readFile();

String anser = "5+6" + "\t" + "11";

String wrilteLase = readFiles.get(readFiles.size() - 1);

readFiles.remove(readFiles.size() - 1);

write(readFiles);

Assert.assertEquals(wrilteLase, anser);

} catch (IOException e){}

}

private List<String> readFile() throws IOException {

return Files.readAllLines(Paths.get("file.txt"), StandardCharsets.UTF\_8);

}

private void write(List<String> readFiles) throws IOException {

try {

BufferedWriter writer = new BufferedWriter(new FileWriter("file.txt"));

for(String str : readFiles) {

writer.write(str);

writer.write(System.getProperty("line.separator"));

}

writer.close();

} catch (IOException e) {

}

}

}

**TravisCINewUserHistoryUITests**

package com.javatechie.travis.api.newTests;

import com.javatechie.travis.api.SimpleGUI;

import org.assertj.swing.edt.GuiActionRunner;

import org.assertj.swing.fixture.FrameFixture;

import org.junit.Assert;

import java.io.BufferedWriter;

import java.io.FileWriter;

import java.io.IOException;

import java.nio.charset.StandardCharsets;

import java.nio.file.Files;

import java.nio.file.Paths;

import java.util.List;

public class TravisCINewUserHistoryUITests {

@org.junit.jupiter.api.Test

void simpleTest() throws IOException {

List<String> content = readFile();

SimpleGUI frame = GuiActionRunner.execute(() -> new SimpleGUI());

FrameFixture window = new FrameFixture(frame);

window.show();

window.textBox().enterText("5+5+1-2");

window.button("actionButton").click();

String[][] contents = window.table().contents();

Assert.assertEquals(contents.length, readFile().size());

write(content);

window.cleanUp();

}

@org.junit.jupiter.api.Test

void simpleTest2() throws IOException {

List<String> content = readFile();

SimpleGUI frame = GuiActionRunner.execute(() -> new SimpleGUI());

FrameFixture window = new FrameFixture(frame);

window.show();

window.textBox().enterText("5+1");

window.button("actionButton").click();

String[][] contents = window.table().contents();

Assert.assertEquals(contents.length, readFile().size());

write(content);

window.cleanUp();

}

private List<String> readFile() throws IOException {

return Files.readAllLines(Paths.get("file.txt"), StandardCharsets.UTF\_8);

}

private void write(List<String> readFiles) throws IOException {

try {

BufferedWriter writer = new BufferedWriter(new FileWriter("file.txt"));

for(String str : readFiles) {

writer.write(str);

writer.write(System.getProperty("line.separator"));

}

writer.close();

} catch (IOException e) {

}

}

}

# ПРИЛОЖЕНИЕ B

**Листинг файла для непрерывной интеграции**

|  |
| --- |
| language: java |
|  | jdk: |
|  | - openjdk8 |
|  | https://travis-ci.org/dist: xenial |
|  | services: |
|  | - xvfb |
|  | script: ./mvnw clean install |

# ПРИЛОЖЕНИЕ C

**УКПО КУРСОВАЯ РАБОТА**