**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ**

**УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 4

по дисциплине:

«Введение в разработку программного обеспечения**»**

«Инструменты программирования и отладки. Системы контроля версий»

Выполнил: студент гр. ИТП-21

Мартынов В.Ю.

Принял: преподаватель-стажер

Башаримов Ю.С.

Гомель 2024

**Цель работы**: изучить инструменты программирования и отладки, системы контроля версий. Научиться работать с системой контроля версий *Git.*

**Ход работы**

Задание 1. Создать локальный git репозиторий. Затем создать три ветки (main, lr1, lr3). Закоммитить лабораторные работы 1 и 3 в ветки lr1 и lr3 соответственно. После коммита создать удаленный репозиторий на GitHub. Привязать локальный репозиторий к удаленному. Запушить ветки в удаленный репозиторий. Создать и одобрить пул-реквесты на слияние веток lr1 и lr3 с веткой main.

Для создания локального репозитория используется команда *git init.* Процесс и результат создания локального репозиторию описан на рисунке 1

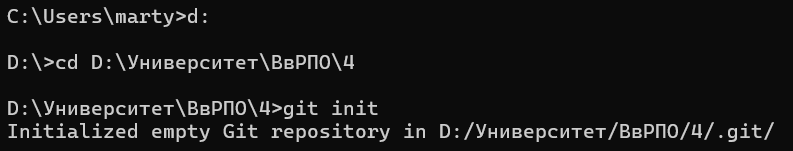


Рисунок 1 –Создание локального репозитория

После выполнения команды локальный репозиторий появтися в заданном каталоге, как показано на рисунке 2.

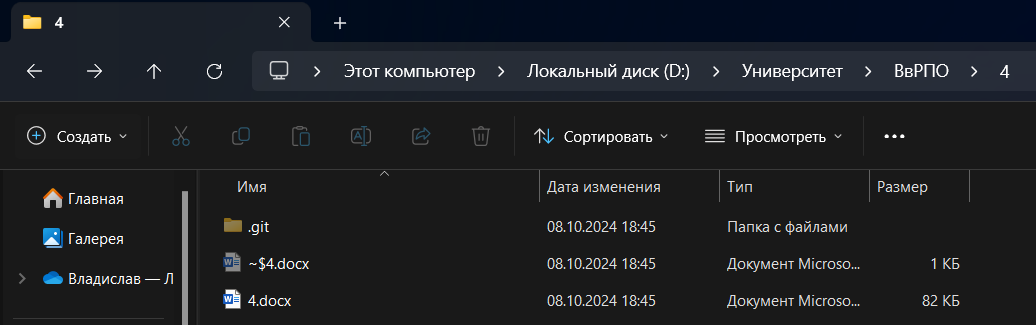


Рисунок 3 –Создание локального репозитория

Для того, чтобы в git создать ветку main, необходимо туда что-то закоммитить. Для этого используются соответствующие команды git add . и git commit -m "Initial commit", как показано на рисунке 2.

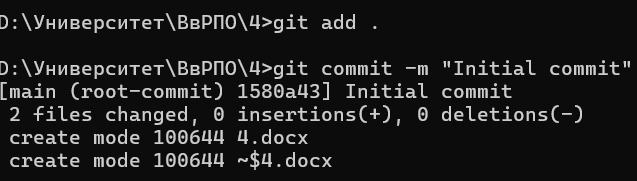


Рисунок 2 –Создание ветки main

После выполнения команд создастся ветка main в папке refs, как показано на рисунке 3.

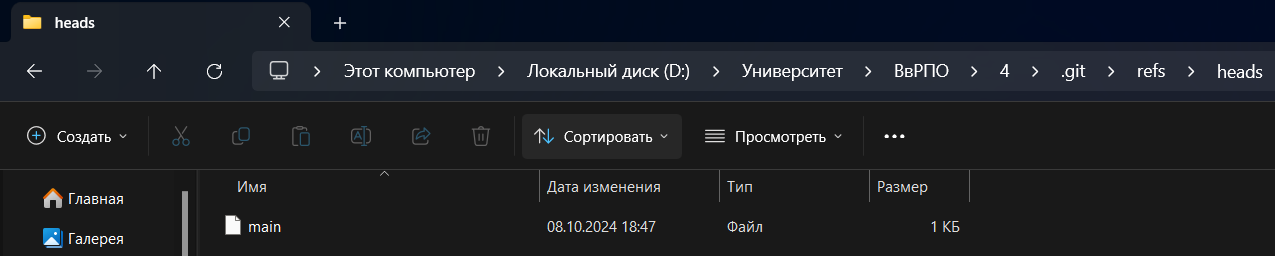


Рисунок 3 –Результат создания ветки main

Далее необходимо создать ветки lr1 и lr3. Для создания используется команда git checkout -b <имя ветки>. После выполнения ключа -b создается ветка с заданным именем и происходит переход на данную ветку за счёт команды checkout. Стоит добавить, что для создания веток можно использовать также команду git branch, однако переход на созданную ветку выполнен не будет. Результат создания веток lr1 и lr3 показан на рисунке 4.

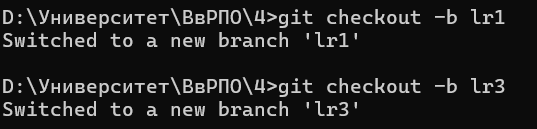


Рисунок 4 –Создание веток lr1 и lr3

После создания ветки появятся в одной директории с веткой main, как показано на рисунке 5.

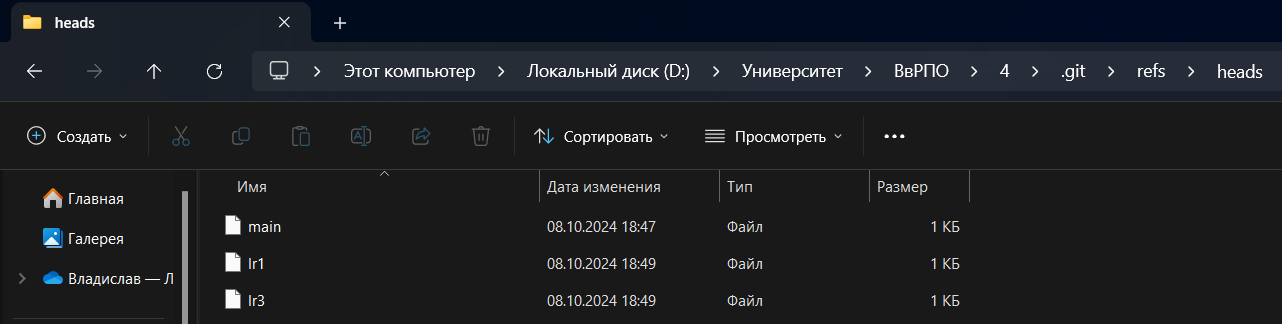
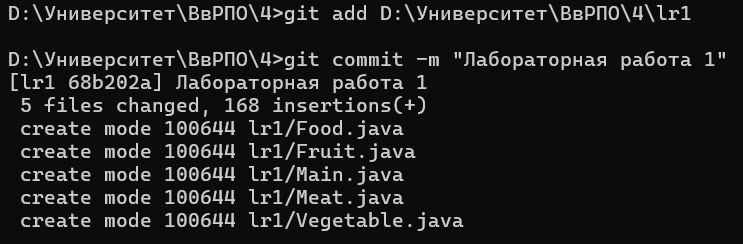
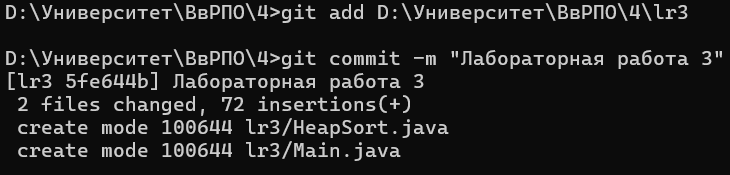
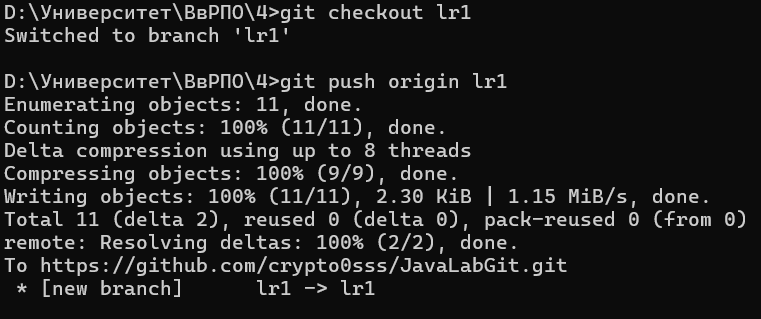


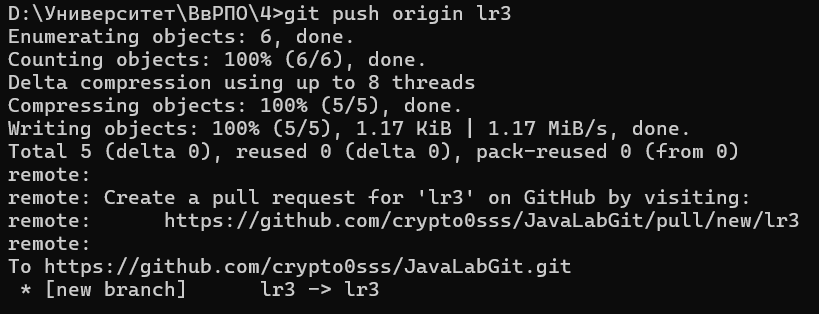
Рисунок 5 –Созданные ветки











**Вывод**: в ходе данной лабораторной работы

**ПРИЛОЖЕНИЕ А**

(обязательное)

**Листинг программы**

**Main.java**

import FoodLib.Food;

import FoodLib.Fruit;

import FoodLib.Meat;

import FoodLib.Vegetable;

import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

List<Food> foodList = new ArrayList<>();

List<String> foodNames = new ArrayList<>();

//Курица

Map<String, Double> chickenNutritionalValue = new HashMap<>();

chickenNutritionalValue.put("Белки", 20.8);

chickenNutritionalValue.put("Жиры", 8.8);

chickenNutritionalValue.put("Углеводы", 0.6);

foodList.add(new Meat("Курица", true, chickenNutritionalValue));

//Индейка

Map<String, Double> turkeyNutritionalValue = new HashMap<>();

turkeyNutritionalValue.put("Белки", 21.6);

turkeyNutritionalValue.put("Жиры", 12.0);

turkeyNutritionalValue.put("Углеводы", 0.8);

foodList.add(new Meat("Индейка", true, turkeyNutritionalValue));

//Баранина

Map<String, Double> muttonNutritionalValue = new HashMap<>();

muttonNutritionalValue.put("Белки", 18.2);

muttonNutritionalValue.put("Жиры", 15.3);

muttonNutritionalValue.put("Углеводы", 0.1);

foodList.add(new Meat("Баранина", false, muttonNutritionalValue));

//Яблоко

Map<String, Double> appleNutritionalValue = new HashMap<>();

appleNutritionalValue.put("Белки", 0.5);

appleNutritionalValue.put("Жиры", 0.3);

appleNutritionalValue.put("Углеводы", 10.0);

foodList.add(new Fruit("Яблоко", false, appleNutritionalValue));

//Апельсин

Map<String, Double> orangeNutritionalValue = new HashMap<>();

orangeNutritionalValue.put("Белки", 1.0);

orangeNutritionalValue.put("Жиры", 0.2);

orangeNutritionalValue.put("Углеводы", 15.0);

foodList.add(new Fruit("Апельсин", true, orangeNutritionalValue));

//Груша

Map<String, Double> pearNutritionalValue = new HashMap<>();

pearNutritionalValue.put("Белки", 0.6);

pearNutritionalValue.put("Жиры", 0.4);

pearNutritionalValue.put("Углеводы", 12.0);

foodList.add(new Fruit("Груша", true, pearNutritionalValue));

//Помидор

Map<String, Double> tomatoNutritionalValue = new HashMap<>();

tomatoNutritionalValue.put("Белки", 1.1);

tomatoNutritionalValue.put("Жиры", 0.2);

tomatoNutritionalValue.put("Углеводы", 3.8);

foodList.add(new Vegetable("Помидор", true, tomatoNutritionalValue));

//Морковь

Map<String, Double> carrotNutritionalValue = new HashMap<>();

carrotNutritionalValue.put("Белки", 0.9);

carrotNutritionalValue.put("Жиры", 0.2);

carrotNutritionalValue.put("Углеводы", 0.2);

foodList.add(new Vegetable("Помидор", true, carrotNutritionalValue));

//Картофель

Map<String, Double> potatoNutritionalValue = new HashMap<>();

potatoNutritionalValue.put("Белки", 2.0);

potatoNutritionalValue.put("Жиры", 0.4);

potatoNutritionalValue.put("Углеводы", 16.3);

foodList.add(new Vegetable("Картофель", true, potatoNutritionalValue));

for (Food food : foodList) {

System.out.println("Создан объект " + food.getName());

System.out.println(food);

foodNames.add(food.getName());

}

System.out.print("Введите букву, на которое оканчивается название еды: ");

char targetLetter = sc.nextLine().charAt(0);

for (String food : foodNames) {

if (food.toLowerCase().endsWith(String.valueOf(targetLetter))) {

System.out.println(food);

}

}

}

}

**Food.java**

package FoodLib;

import java.util.Map;

public abstract class Food {

private final boolean edible;

private final Map<String, Double> nutritionalValue;

private final String name;

public Food(boolean edible, Map<String, Double> nutritionalValue, String name) {

this.edible = edible;

this.nutritionalValue = nutritionalValue;

this.name = name;

}

private double calculateEnergyValue() {

double protein = nutritionalValue.getOrDefault("Белки", 0.0);

double fat = getNutritionalValue().getOrDefault("Жиры", 0.0);

double carbs = getNutritionalValue().getOrDefault("Углеводы", 0.0);

return 4 \* protein + 9 \* fat + 4 \* carbs;

}

public boolean isEdible() {

return edible;

}

public Map<String, Double> getNutritionalValue() {

return nutritionalValue;

}

public String getName() {

return name;

}

@Override

public String toString() {

return " " + name + "\n Съедобно:" + (edible ? "Да" : "Нет") + "\n Энергетическая ценность "+ name + " : " + calculateEnergyValue();

}

}

**Meat.java**

import java.util.Map;

class Meat extends Food {

public Meat(String name, boolean edible, Map<String, Double> nutritionalValue) {

super(edible, nutritionalValue, name);

}

}

**Fruit.java**

import java.util.Map;

class Fruit extends Food {

public Fruit(String name, boolean edible, Map<String, Double> nutritionalValue) {

super(edible, nutritionalValue, name);

}

}

**Vegetable.java**

import java.util.Map;

class Vegetable extends Food {

public Vegetable(String name, boolean edible, Map<String, Double> nutritionalValue) {

super(edible, nutritionalValue, name);

}

}