Міністерство освіти і науки України

Національний університет «Львівська Політехніка»

Інститут комп’ютерних наук та інформаційних технологій

**ЛАБОРАТОРНА РОБОТА№4**

З дисципліни **«**Прикладне програмування**»**

Тема:

**«Комплексна ЛР»**

|  |
| --- |
| **Виконав:** |
| **Грева Олег Олегович** |
| **Група:**  **КН-204** |

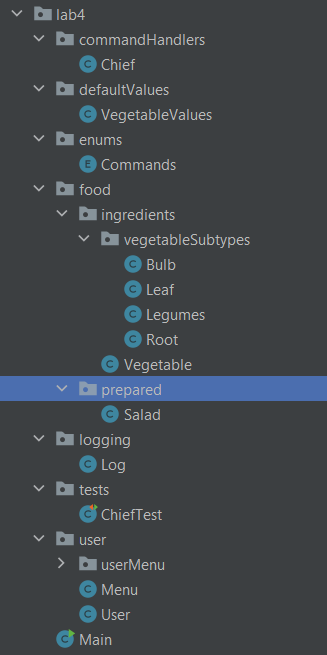
|  |
| --- |
| **Прийняв:** |
| **Вергун В.Р.** |
|  |

Львів 2020

**Індивідуальне завдання**

Шеф-повар. Визначити ієрархію овочів. Зробити салат. Підрахувати калорійність. Здійснити сортування овочів для салату на основі одного з параметрів. Знайти овочі в салаті, що відповідають заданому діапазону калорійності.

**Класи та пакети**

****

**Тестування класу Chief**

class ChiefTest {  
  
 @Test  
 void addUnExistingIngredient() {  
 Salad salad = new Salad();  
  
 Assertions.*assertThrows*(IllegalStateException.class ,() -> {  
 Chief.addIngredient("peanut",salad);  
 });  
 }

@Test  
 void addIngredientAlreadyInSalad() {  
 Salad salad = new Salad();  
 Chief.addIngredient("carrot",salad);  
 Assertions.assertThrows(IllegalStateException.class ,() -> {  
 Chief.addIngredient("carrot",salad);  
 });  
 }  
 @Test  
 void removeIngredientNotInSalad() {  
 Salad salad = new Salad();  
  
 Assertions.*assertThrows*(IllegalStateException.class ,() -> {  
 Chief.*removeIngredient*("carrot",salad);  
 });  
 }  
  
 @Test  
 void findByCaloriesSwappedBorders() {  
 Salad salad = new Salad();  
  
 Assertions.assertThrows(IllegalStateException.class ,() -> {  
 Chief.findByCalories(12,10,salad);  
 });  
 }  
  
 @Test  
 void countCaloriesInEmptySaladReturnZero() {  
 Salad salad = new Salad();  
  
 double result = Chief.countCalories(salad);  
 Assertions.assertEquals(0,result);  
 }

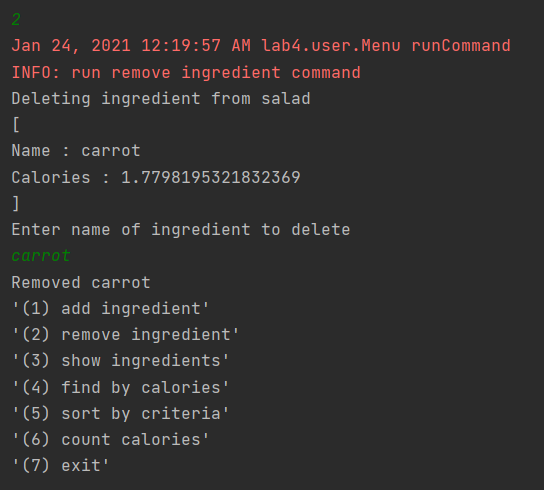
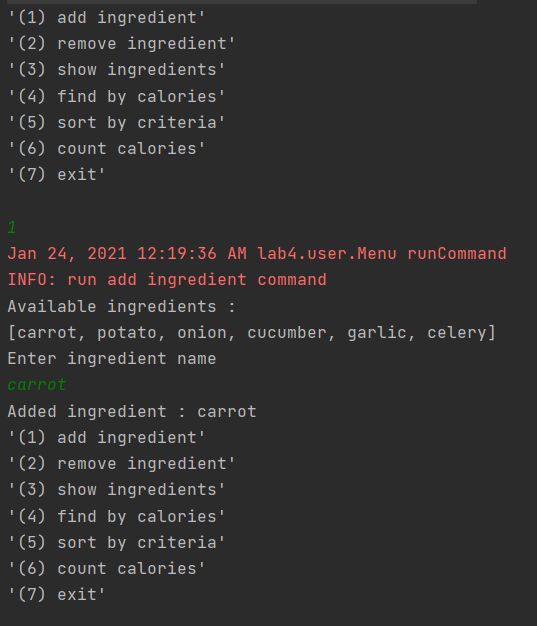
**Логгер**

public class Log {  
  
 private Logger logger;  
 private FileHandler handler;  
  
 public Log(String fileName) throws SecurityException, IOException {  
  
 File file = new File(fileName);  
  
 if(!file.exists()) {  
 file.createNewFile();  
 }  
  
 handler = new FileHandler(fileName,true);  
 logger = Logger.*getLogger*("myLogger");  
 logger.addHandler(handler);  
 handler.setFormatter(new SimpleFormatter());  
 }  
 public Logger getLogger() {  
 return logger;  
 }  
}

**Головне меню**

public class Menu {  
 private Log log;  
 {  
 try {  
 log = new Log("myLog.txt");  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 private final Scanner userInput = new Scanner(System.*in*);  
  
 Salad salad = new Salad();  
 User user = new User();  
  
 public Menu() {  
  
 user.addCommand("add ingredient",new AddIngredientCommand(salad));  
 user.addCommand("count calories",new CountCaloriesCommand(salad));  
 user.addCommand("find by calories",new FindByCaloriesCommand(salad));  
 user.addCommand("remove ingredient",new RemoveIngredientCommand(salad));  
 user.addCommand("show ingredients",new ShowIngredientsCommand(salad));  
 user.addCommand("sort by criteria",new SortByCriteriaCommand(salad));  
 }  
  
 public void execute() {  
  
 log.getLogger().setLevel(Level.*FINEST*);  
  
 while(true) {  
  
 showMenu();  
 runCommand(getCommand());  
 }  
 }  
  
 private void runCommand(Commands command) {  
  
 switch(command) {  
  
 case *ADD\_INGREDIENT*: {  
 log.getLogger().info("run add ingredient command");  
 user.runCommand("add ingredient");  
 break;  
 }  
  
 case *REMOVE\_INGREDIENT*: {  
 log.getLogger().info("run remove ingredient command");  
 user.runCommand("remove ingredient");  
 break;  
 }  
  
 case *SHOW\_INGREDIENTS*: {  
 log.getLogger().info("run show ingredients command");  
 user.runCommand("show ingredients");  
 break;  
 }  
  
 case *FIND\_BY\_CALORIES*: {  
 log.getLogger().info("run find by calories command");  
 user.runCommand("find by calories");  
 break;  
 }  
  
 case *SORT\_BY\_CRITERIA*: {  
 log.getLogger().info("run sort by criteria command");  
 user.runCommand("sort by criteria");  
 break;  
 }  
  
 case *COUNT\_CALORIES*: {  
 log.getLogger().info("run count calories command");  
 user.runCommand("count calories");  
 break;  
 }  
  
 case *EXIT*: {  
 log.getLogger().info("run exit command");  
 return;  
 }  
  
 default: {  
 log.getLogger().warning("given wrong input in main menu");  
 System.*out*.println("Wrong input");  
 }  
 }  
 }  
  
 private void showMenu() {  
  
 System.*out*.println("'(1) add ingredient'\n" +  
 "'(2) remove ingredient'\n" +  
 "'(3) show ingredients'\n" +  
 "'(4) find by calories'\n" +  
 "'(5) sort by criteria'\n" +  
 "'(6) count calories'\n" +  
 "'(7) exit'\n");  
 }  
  
 private Commands getCommand() {  
  
 int command = userInput.nextInt();  
 return Commands.*values*()[command-1];  
 }  
}

**Результат виконання програми**

****