**РУСЕНСКИ УНИВЕРСИТЕТ “АНГЕЛ КЪНЧЕВ”**

КУРСОВА РАБОТА  
ПО ПРОГРАМНИ ЕЗИЦИ

Студент:

Факултетен номер:

Група:

Специалност:

Дата: Изготвил:  
 Проверил:

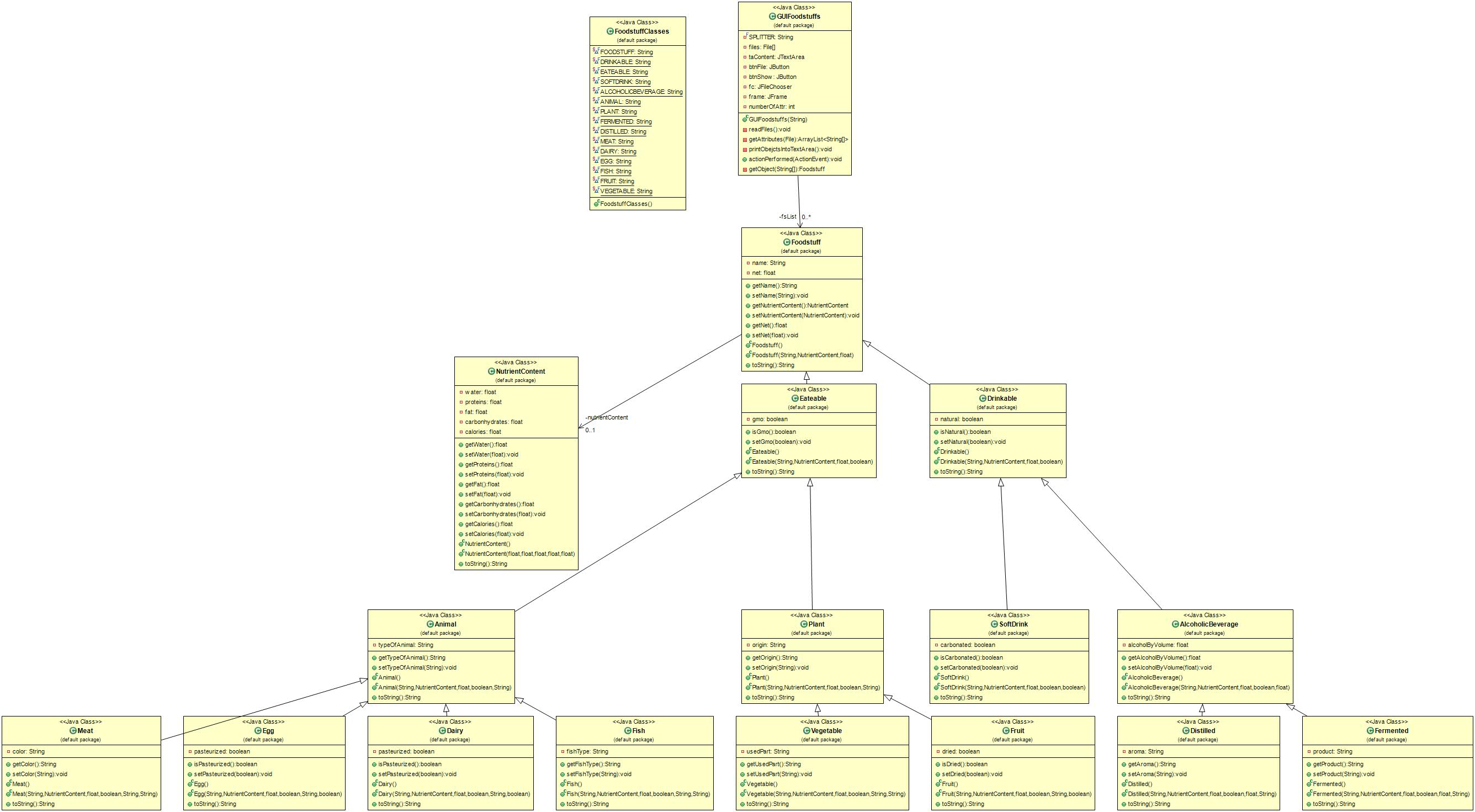
# Задание.

Да се състави йерархия от класове, описващи **хранителни продукти** и програма на Java с графичен потреителски интерфейс, включваща следните функционалности:

* въвеждане на данни за обекти от тези класове;
* съхранение на данните;

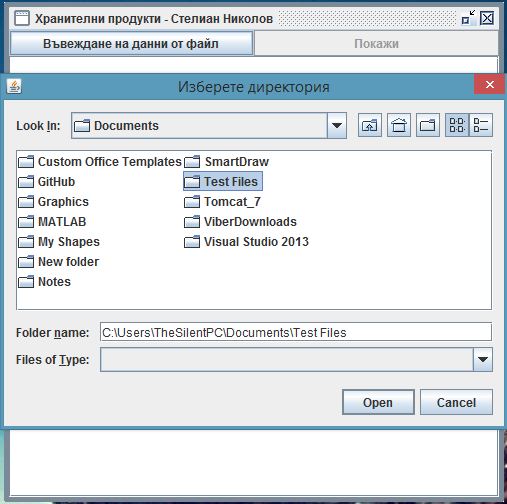
Данните за обектите да се попълват от **текстов файл, като данните за обектите на всеки клас са в отделни файлове (т.е. има толкова файлове, колкото са класовете в йерархията) (форматът им е по избор на студента)** и да се съхраняват в **ArrayList (java.util.ArrayList)**. Изборът на атрибути и методи е на студента, но броят, типовете им и разположението име в йерархията влияят на оценката.

# Клас диаграми.

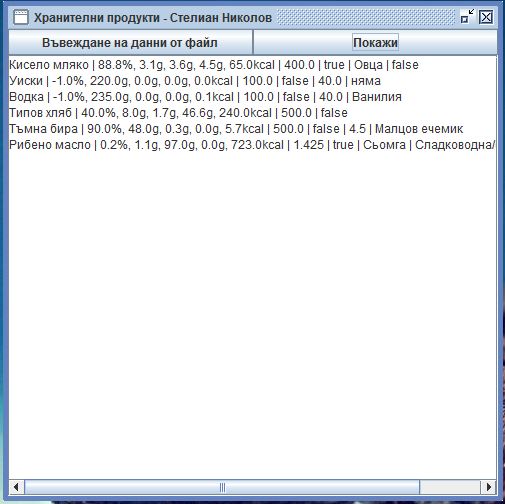


Фигура 1 Клас диаграма.

# Тестови примери



Фигура 2 Избор на файл



Фигура 3 Визуализиране на данните

# Листинг.

ALCOHOLICBEVERAGE.JAVA

public class AlcoholicBeverage extends Drinkable {

private float alcoholByVolume;

public float getAlcoholByVolume() {

return alcoholByVolume;

}

public void setAlcoholByVolume(float alcoholByVolume) {

this.alcoholByVolume = alcoholByVolume;

}

public AlcoholicBeverage() {

}

public AlcoholicBeverage(String name, NutrientContent nutrientContent,

float net, boolean natural, float alcoholByVolume) {

super(name, nutrientContent, net, natural);

this.alcoholByVolume = alcoholByVolume;

}

@Override

public String toString() {

StringBuilder builder = new StringBuilder();

builder.append(super.toString()).append(" | ");

builder.append(alcoholByVolume);

return builder.toString();

}

}

ANIMAL.JAVA

public class Animal extends Eateable {

private String typeOfAnimal;

public String getTypeOfAnimal() {

return typeOfAnimal;

}

public void setTypeOfAnimal(String typeOfAnimal) {

this.typeOfAnimal = typeOfAnimal;

}

public Animal() {

}

public Animal(String name, NutrientContent nutrientContent, float net,

boolean gmo, String typeOfAnimal) {

super(name, nutrientContent, net, gmo);

this.typeOfAnimal = typeOfAnimal;

}

@Override

public String toString() {

StringBuilder builder = new StringBuilder();

builder.append(super.toString()).append(" | ");

builder.append(typeOfAnimal);

return builder.toString();

}

}

DAIRY.JAVA

public class Dairy extends Animal {

private boolean pasteurized;

public boolean isPasteurized() {

return pasteurized;

}

public void setPasteurized(boolean pasteurized) {

this.pasteurized = pasteurized;

}

public Dairy() {

}

public Dairy(String name, NutrientContent nutrientContent, float net,

boolean gmo, String typeOfAnimal, boolean pasteurized) {

super(name, nutrientContent, net, gmo, typeOfAnimal);

this.pasteurized = pasteurized;

}

@Override

public String toString() {

StringBuilder builder = new StringBuilder();

builder.append(super.toString()).append(" | ");

builder.append(pasteurized);

return builder.toString();

}

}

DISTILLED.JAVA

public class Distilled extends AlcoholicBeverage {

private String aroma;

public String getAroma() {

return aroma;

}

public void setAroma(String aroma) {

this.aroma = aroma;

}

public Distilled() {

}

public Distilled(String name, NutrientContent nutrientContent, float net,

boolean natural, float alcoholByVolume, String aroma) {

super(name, nutrientContent, net, natural, alcoholByVolume);

this.aroma = aroma;

}

@Override

public String toString() {

StringBuilder builder = new StringBuilder();

builder.append(super.toString()).append(" | ");

builder.append(aroma);

return builder.toString();

}

}

DRINKABLE.JAVA

public class Drinkable extends Foodstuff {

private boolean natural;

public boolean isNatural() {

return natural;

}

public void setNatural(boolean natural) {

this.natural = natural;

}

public Drinkable() {

}

public Drinkable(String name, NutrientContent nutrientContent,

float net, boolean natural) {

super(name, nutrientContent, net);

this.natural = natural;

}

@Override

public String toString() {

StringBuilder builder = new StringBuilder();

builder.append(super.toString()).append(" | ");

builder.append(natural);

return builder.toString();

}

}

EATEABLE.JAVA

public class Eateable extends Foodstuff {

private boolean gmo;

public boolean isGmo() {

return gmo;

}

public void setGmo(boolean gmo) {

this.gmo = gmo;

}

public Eateable() {

}

public Eateable(String name, NutrientContent nutrientContent,

float net, boolean gmo) {

super(name, nutrientContent, net);

this.gmo = gmo;

}

@Override

public String toString() {

StringBuilder builder = new StringBuilder();

builder.append(super.toString()).append(" | ");

builder.append(gmo);

return builder.toString();

}

}

EGG.JAVA

public class Egg extends Animal {

private boolean pasteurized;

public boolean isPasteurized() {

return pasteurized;

}

public void setPasteurized(boolean pasteurized) {

this.pasteurized = pasteurized;

}

public Egg() {

}

public Egg(String name, NutrientContent nutrientContent, float net,

boolean gmo, String typeOfAnimal, boolean pasteurized) {

super(name, nutrientContent, net, gmo, typeOfAnimal);

this.pasteurized = pasteurized;

}

@Override

public String toString() {

StringBuilder builder = new StringBuilder();

builder.append(super.toString()).append(" | ");

builder.append(pasteurized);

return builder.toString();

}

}

FERMENTED.JAVA

public class Fermented extends AlcoholicBeverage {

private String product;

public String getProduct() {

return product;

}

public void setProduct(String product) {

this.product = product;

}

public Fermented() {

}

public Fermented(String name, NutrientContent nutrientContent, float net,

boolean natural, float alcoholByVolume, String product) {

super(name, nutrientContent, net, natural, alcoholByVolume);

this.product = product;

}

@Override

public String toString() {

StringBuilder builder = new StringBuilder();

builder.append(super.toString()).append(" | ");

builder.append(product);

return builder.toString();

}

}

FISH.JAVA

public class Fish extends Animal {

private String fishType;

public String getFishType() {

return fishType;

}

public void setFishType(String fishType) {

this.fishType = fishType;

}

public Fish() {

}

public Fish(String name, NutrientContent nutrientContent, float net,

boolean gmo, String typeOfAnimal, String fishType) {

super(name, nutrientContent, net, gmo, typeOfAnimal);

this.fishType = fishType;

}

@Override

public String toString() {

StringBuilder builder = new StringBuilder();

builder.append(super.toString()).append(" | ");

builder.append(fishType);

return builder.toString();

}

}

FOODSTUFF.JAVA

public class Foodstuff {

private String name;

private NutrientContent nutrientContent;

private float net;

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public NutrientContent getNutrientContent() {

return nutrientContent;

}

public void setNutrientContent(NutrientContent nutrientContent) {

this.nutrientContent = nutrientContent;

}

public float getNet() {

return net;

}

public void setNet(float net) {

this.net = net;

}

public Foodstuff() {

}

public Foodstuff(String name, NutrientContent nutrientContent, float net) {

super();

this.name = name;

this.nutrientContent = nutrientContent;

this.net = net;

}

@Override

public String toString() {

StringBuilder builder = new StringBuilder();

builder.append("\n").append(name).append(" | ");

builder.append(nutrientContent).append(" | ");

builder.append(net);

return builder.toString();

}

}

FOODSTUFFCLASSES.JAVA

public class FoodstuffClasses {

static final String FOODSTUFF = "foodstuff";

static final String DRINKABLE = "drinkable";

static final String EATEABLE = "eateable";

static final String SOFTDRINK = "softdrink";

static final String ALCOHOLICBEVERAGE = "alcoholicbeverage";

static final String ANIMAL = "animal";

static final String PLANT = "plant";

static final String FERMENTED = "fermented";

static final String DISTILLED = "distilled";

static final String MEAT = "meat";

static final String DAIRY = "dairy";

static final String EGG = "egg";

static final String FISH = "fish";

static final String FRUIT = "fruit";

static final String VEGETABLE = "vegetable";

}

FRUIT.JAVA

public class Fruit extends Plant {

private boolean dried;

public boolean isDried() {

return dried;

}

public void setDried(boolean dried) {

this.dried = dried;

}

public Fruit() {

}

public Fruit(String name, NutrientContent nutrientContent, float net,

boolean gmo, String origin, boolean dried) {

super(name, nutrientContent, net, gmo, origin);

this.dried = dried;

}

@Override

public String toString() {

StringBuilder builder = new StringBuilder();

builder.append(super.toString()).append(" | ");

builder.append(dried);

return builder.toString();

}

}

GUIFOODSTUFFS.JAVA

import java.awt.BorderLayout;

import java.awt.GridLayout;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.io.File;

import java.io.FileNotFoundException;

import java.io.FilenameFilter;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.Scanner;

import javax.swing.JButton;

import javax.swing.JFileChooser;

import javax.swing.JFrame;

import javax.swing.JPanel;

import javax.swing.JScrollPane;

import javax.swing.JTextArea;

public class GUIFoodstuffs implements ActionListener {

private final String SPLITTER = ":";

private File[] files;

private JTextArea taContent;

private JButton btnFile;

private JButton btnShow;

private JFileChooser fc;

private JFrame frame;

private ArrayList<Foodstuff> fsList;

private int numberOfAttr;

public GUIFoodstuffs(String title) {

fsList = new ArrayList<>();

JFrame.setDefaultLookAndFeelDecorated(true);

frame = new JFrame(title);

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

frame.getContentPane().setLayout(new BorderLayout());

fc = new JFileChooser();

fc.setCurrentDirectory(new java.io.File("."));

fc.setDialogTitle("Изберете директория");

fc.setFileSelectionMode(JFileChooser.DIRECTORIES\_ONLY);

fc.setAcceptAllFileFilterUsed(false);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

JPanel pnlButtons = new JPanel(new GridLayout(0, 2));

btnFile = new JButton("Въвеждане на данни от файл");

btnFile.addActionListener(this);

pnlButtons.add(btnFile);

btnShow = new JButton("Покажи");

btnShow.addActionListener(this);

btnShow.setEnabled(false);

pnlButtons.add(btnShow);

frame.getContentPane().add(pnlButtons, BorderLayout.PAGE\_START);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

JPanel pnlContent = new JPanel(new BorderLayout());

taContent = new JTextArea();

JScrollPane scroll = new JScrollPane (taContent,

JScrollPane.VERTICAL\_SCROLLBAR\_AS\_NEEDED, JScrollPane.HORIZONTAL\_SCROLLBAR\_AS\_NEEDED);

pnlContent.add(scroll, BorderLayout.CENTER);

frame.getContentPane().add(pnlContent, BorderLayout.CENTER);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

frame.pack();

frame.setSize(500, 500);

frame.setResizable(false);

frame.setLocationRelativeTo(null);

frame.setVisible(true);

}

private void readFiles() {

fsList.clear();

numberOfAttr = Foodstuff.class.getDeclaredFields().length - 1; // minus 1 NutrientContent

numberOfAttr += NutrientContent.class.getDeclaredFields().length;

for (File file : files) {

ArrayList<String[]> objectsAttr = getAttributes(file);

for (String[] item : objectsAttr) {

Foodstuff foodstuff =

getObject(Arrays.copyOfRange(item, 0, numberOfAttr));

String fileName = file.getName().substring(0, file.getName().

length() - ".txt".length()).toLowerCase();

switch (fileName) {

case FoodstuffClasses.ALCOHOLICBEVERAGE:

AlcoholicBeverage ab = new AlcoholicBeverage();

ab.setName(foodstuff.getName());

ab.setNutrientContent(foodstuff.getNutrientContent());

ab.setNet(foodstuff.getNet());

ab.setNatural(Boolean.parseBoolean(item[numberOfAttr]));

ab.setAlcoholByVolume(Float.parseFloat(item[numberOfAttr+1]));

fsList.add(ab);

break;

case FoodstuffClasses.ANIMAL:

Animal animal = new Animal();

animal.setName(foodstuff.getName());

animal.setNutrientContent(foodstuff.getNutrientContent());

animal.setNet(foodstuff.getNet());

animal.setGmo(Boolean.parseBoolean(item[numberOfAttr]));

animal.setTypeOfAnimal(item[numberOfAttr+1]);

fsList.add(animal);

break;

case FoodstuffClasses.DAIRY:

Dairy dairy = new Dairy();

dairy.setName(foodstuff.getName());

dairy.setNutrientContent(foodstuff.getNutrientContent());

dairy.setNet(foodstuff.getNet());

dairy.setGmo(Boolean.parseBoolean(item[numberOfAttr]));

dairy.setTypeOfAnimal(item[numberOfAttr+1]);

dairy.setPasteurized(Boolean.parseBoolean(item[3]));

fsList.add(dairy);

break;

case FoodstuffClasses.DISTILLED:

Distilled distilled = new Distilled();

distilled.setName(foodstuff.getName());

distilled.setNutrientContent(foodstuff.getNutrientContent());

distilled.setNet(foodstuff.getNet());

distilled.setNatural(Boolean.parseBoolean(item[numberOfAttr]));

distilled.setAlcoholByVolume(Float.parseFloat(item[numberOfAttr+1]));

distilled.setAroma(item[numberOfAttr+2]);

fsList.add(distilled);

break;

case FoodstuffClasses.DRINKABLE:

Drinkable drinkable = new Drinkable();

drinkable.setName(foodstuff.getName());

drinkable.setNutrientContent(foodstuff.getNutrientContent());

drinkable.setNet(foodstuff.getNet());

drinkable.setNatural(Boolean.parseBoolean(item[numberOfAttr]));

fsList.add(drinkable);

break;

case FoodstuffClasses.EATEABLE:

Eateable eatable = new Eateable();

eatable.setName(foodstuff.getName());

eatable.setNutrientContent(foodstuff.getNutrientContent());

eatable.setNet(foodstuff.getNet());

eatable.setGmo(Boolean.parseBoolean(item[numberOfAttr]));

fsList.add(eatable);

break;

case FoodstuffClasses.EGG:

Egg egg = new Egg();

egg.setName(foodstuff.getName());

egg.setNutrientContent(foodstuff.getNutrientContent());

egg.setNet(foodstuff.getNet());

egg.setGmo(Boolean.parseBoolean(item[numberOfAttr]));

egg.setTypeOfAnimal(item[numberOfAttr+1]);

egg.setPasteurized(Boolean.parseBoolean(item[3]));

fsList.add(egg);

break;

case FoodstuffClasses.FERMENTED:

Fermented fermented = new Fermented();

fermented.setName(foodstuff.getName());

fermented.setNutrientContent(foodstuff.getNutrientContent());

fermented.setNet(foodstuff.getNet());

fermented.setNatural(Boolean.parseBoolean(item[numberOfAttr]));

fermented.setAlcoholByVolume(Float.parseFloat(item[numberOfAttr+1]));

fermented.setProduct(item[numberOfAttr+2]);

fsList.add(fermented);

break;

case FoodstuffClasses.FISH:

Fish fish = new Fish();

fish.setName(foodstuff.getName());

fish.setNutrientContent(foodstuff.getNutrientContent());

fish.setNet(foodstuff.getNet());

fish.setGmo(Boolean.parseBoolean(item[numberOfAttr]));

fish.setTypeOfAnimal(item[numberOfAttr+1]);

fish.setFishType(item[numberOfAttr+2]);

fsList.add(fish);

break;

case FoodstuffClasses.FOODSTUFF:

fsList.add(foodstuff);

break;

case FoodstuffClasses.FRUIT:

Fruit fruit = new Fruit();

fruit.setName(foodstuff.getName());

fruit.setNutrientContent(foodstuff.getNutrientContent());

fruit.setNet(foodstuff.getNet());

fruit.setGmo(Boolean.parseBoolean(item[numberOfAttr]));

fruit.setOrigin(item[numberOfAttr+1]);

fruit.setDried(Boolean.parseBoolean(item[numberOfAttr+2]));

fsList.add(fruit);

break;

case FoodstuffClasses.MEAT:

Meat meat = new Meat();

meat.setName(foodstuff.getName());

meat.setNutrientContent(foodstuff.getNutrientContent());

meat.setNet(foodstuff.getNet());

meat.setGmo(Boolean.parseBoolean(item[numberOfAttr]));

meat.setTypeOfAnimal(item[numberOfAttr+1]);

meat.setColor(item[numberOfAttr+2]);

fsList.add(meat);

break;

case FoodstuffClasses.PLANT:

Plant plant = new Plant();

plant.setName(foodstuff.getName());

plant.setNutrientContent(foodstuff.getNutrientContent());

plant.setNet(foodstuff.getNet());

plant.setGmo(Boolean.parseBoolean(item[numberOfAttr]));

plant.setOrigin(item[numberOfAttr+1]);

fsList.add(plant);

break;

case FoodstuffClasses.SOFTDRINK:

SoftDrink softDrink = new SoftDrink();

softDrink.setName(foodstuff.getName());

softDrink.setNutrientContent(foodstuff.getNutrientContent());

softDrink.setNet(foodstuff.getNet());

softDrink.setNatural(Boolean.parseBoolean(item[numberOfAttr]));

softDrink.setCarbonated(Boolean.parseBoolean(item[numberOfAttr+1]));

fsList.add(softDrink);

break;

case FoodstuffClasses.VEGETABLE:

Vegetable vegetable = new Vegetable();

vegetable.setName(foodstuff.getName());

vegetable.setNutrientContent(foodstuff.getNutrientContent());

vegetable.setNet(foodstuff.getNet());

vegetable.setGmo(Boolean.parseBoolean(item[numberOfAttr]));

vegetable.setOrigin(item[numberOfAttr+1]);

vegetable.setUsedPart(item[numberOfAttr+2]);

fsList.add(vegetable);

break;

}

}

}

}

@SuppressWarnings("resource")

private ArrayList<String[]> getAttributes(File file) {

ArrayList<String[]> result = new ArrayList<>();

Scanner fileScanner;

try {

fileScanner = new Scanner(file);

while (fileScanner.hasNextLine()) {

result.add(fileScanner.nextLine().split(SPLITTER));

}

return result;

} catch (FileNotFoundException e) {

e.printStackTrace();

}

return null;

}

private void printObejctsIntoTextArea() {

taContent.setText("");

for (Foodstuff f : fsList) {

taContent.append(f.toString());

}

taContent.setText(taContent.getText().trim());

}

@Override

public void actionPerformed(ActionEvent e) {

if (e.getSource() == btnFile) {

int returnVal = fc.showOpenDialog(frame);

if (returnVal == JFileChooser.APPROVE\_OPTION) {

files = fc.getSelectedFile().listFiles(new FilenameFilter() {

public boolean accept(File dir, String name) {

return name.toLowerCase().endsWith(".txt");

}

});

btnShow.setEnabled(true);

readFiles();

} else if (files == null) {

btnShow.setEnabled(false);

}

} else if (e.getSource() == btnShow) {

printObejctsIntoTextArea();

}

}

private Foodstuff getObject(String[] attributes) {

if (attributes != null && attributes.length == numberOfAttr) {

Foodstuff obj = new Foodstuff();

obj.setName(attributes[0]);

NutrientContent nutrientContent = new NutrientContent();

nutrientContent.setWater(Float.parseFloat(attributes[1]));

nutrientContent.setProteins(Float.parseFloat(attributes[2]));

nutrientContent.setFat(Float.parseFloat(attributes[3]));

nutrientContent.setCarbonhydrates(Float.parseFloat(attributes[4]));

nutrientContent.setCalories(Float.parseFloat(attributes[5]));

obj.setNutrientContent(nutrientContent);

obj.setNet(Float.parseFloat(attributes[6]));

return obj;

}

return null;

}

}

MAIN.JAVA

public class Main {

public static void main(String[] args) {

GUIFoodstuffs gui = new GUIFoodstuffs("Хранителни продукти - Стелиан Николов");

}

}

MEAT.JAVA

public class Meat extends Animal {

private String color;

public String getColor() {

return color;

}

public void setColor(String color) {

this.color = color;

}

public Meat() {

}

public Meat(String name, NutrientContent nutrientContent, float net,

boolean gmo, String typeOfAnimal, String color) {

super(name, nutrientContent, net, gmo, typeOfAnimal);

this.color = color;

}

@Override

public String toString() {

StringBuilder builder = new StringBuilder();

builder.append(super.toString()).append(" | ");

builder.append(color);

return builder.toString();

}

}

NUTRIENTCONTENT.JAVA

public class NutrientContent {

private float water;

private float proteins;

private float fat;

private float carbonhydrates;

private float calories;

public float getWater() {

return water;

}

public void setWater(float water) {

this.water = water;

}

public float getProteins() {

return proteins;

}

public void setProteins(float proteins) {

this.proteins = proteins;

}

public float getFat() {

return fat;

}

public void setFat(float fat) {

this.fat = fat;

}

public float getCarbonhydrates() {

return carbonhydrates;

}

public void setCarbonhydrates(float carbonhydrates) {

this.carbonhydrates = carbonhydrates;

}

public float getCalories() {

return calories;

}

public void setCalories(float calories) {

this.calories = calories;

}

public NutrientContent() {

}

public NutrientContent(float water, float proteins, float fat,

float carbonhydrates, float calories) {

super();

this.water = water;

this.proteins = proteins;

this.fat = fat;

this.carbonhydrates = carbonhydrates;

this.calories = calories;

}

@Override

public String toString() {

StringBuilder builder = new StringBuilder();

builder.append(water).append("%, ");

builder.append(proteins).append("g, ");

builder.append(fat).append("g, ");

builder.append(carbonhydrates).append("g, ");

builder.append(calories).append("kcal");

return builder.toString();

}

}

PLANT.JAVA

public class Plant extends Eateable {

private String origin;

public String getOrigin() {

return origin;

}

public void setOrigin(String origin) {

this.origin = origin;

}

public Plant() {

}

public Plant(String name, NutrientContent nutrientContent, float net,

boolean gmo, String origin) {

super(name, nutrientContent, net, gmo);

this.origin = origin;

}

@Override

public String toString() {

StringBuilder builder = new StringBuilder();

builder.append(super.toString()).append(" | ");

builder.append(origin);

return builder.toString();

}

}

SOFTDRINK.JAVA

public class SoftDrink extends Drinkable {

private boolean carbonated;

public boolean isCarbonated() {

return carbonated;

}

public void setCarbonated(boolean carbonated) {

this.carbonated = carbonated;

}

public SoftDrink() {

}

public SoftDrink(String name, NutrientContent nutrientContent, float net,

boolean natural, boolean carbonated) {

super(name, nutrientContent, net, natural);

this.carbonated = carbonated;

}

@Override

public String toString() {

StringBuilder builder = new StringBuilder();

builder.append(super.toString()).append(" | ");

builder.append(carbonated);

return builder.toString();

}

}

VEGETABLE.JAVA

public class Vegetable extends Plant {

private String usedPart;

public String getUsedPart() {

return usedPart;

}

public void setUsedPart(String usedPart) {

this.usedPart = usedPart;

}

public Vegetable() {

}

public Vegetable(String name, NutrientContent nutrientContent, float net,

boolean gmo, String origin, String usedPart) {

super(name, nutrientContent, net, gmo, origin);

this.usedPart = usedPart;

}

@Override

public String toString() {

StringBuilder builder = new StringBuilder();

builder.append(super.toString()).append(" | ");

builder.append(usedPart);

return builder.toString();

}

}