OOP THEORY

Author Mr H Peuckert

ALGORITHMS/PSEUDO CODE

Program Theme: Products sold at a Grocery Shop

```
package productsui;

import java.text.DecimalFormat;

// 1.1
public class Product {

    // 1.2
    // FIELDS\PROPERTIES:

    private String prodID;
    private String prodName;
    private String weight;
    private int prodType;
    private double sellPrice;
    private int quantity;

    // 1.3
    // CONSTANTS:

    public static final double VAT = 15;

    public static final int TYPE_FOOD = 1;
    public static final int TYPE_TOILETRIES = 2;
    public static final int TYPE_OTHER = 3;

    // 1.4
    // HELPER METHOD 1:

    private double addVATandMarkUp(double costPrice, double markUp) {

        double pricePlusVAT = 0;

        pricePlusVAT = costPrice + costPrice * VAT / 100;                  // Add VAT to costPrice
        pricePlusVAT = pricePlusVAT + pricePlusVAT * markUp / 100;   // Increase price by markup
%

        DecimalFormat dec = new DecimalFormat("#.##");                      // Change the real
number
        pricePlusVAT = Double.parseDouble(dec.format(pricePlusVAT));    //       to 2 decimal places

        return pricePlusVAT;
    }

    // 1.5
```

```
// CONSTRUCTOR METHOD:
public Product(String pi, String pn, String w, char pt, double costPrice, double markUp, int quantity) {
        prodID = pi;
        prodName = pn;
        weight = w;

        if (pt == 'F') {
            prodType = TYPE_FOOD;
        } else if (pt == 'T') {
            prodType = TYPE_TOILETRIES;
        } else {
            prodType = TYPE_OTHER;
        }

        sellPrice = addVATandMarkUp(costPrice, markUp);

        this.quantity = quantity;
    }

    // 1.6
    // ACCESSOR\GETTER METHODS:
    public String getProdID() {
        return prodID;
    }

    public String getProdName() {
        return prodName;
    }

    public String getWeight() {
        return weight;
    }

    public int getProdType() {
        return prodType;
    }

    public double getSellPrice() {
        return sellPrice;
    }

    public int getQuantity() {
        return quantity;
    }



    // 1.7
    // MUTATOT/SETTER METHODS:
    public void setProdID(String pi) {
```

```
    prodID = pi;
}

public void setProdName(String pn) {
    prodName = pn;
}

public void setWeight(String w) {
    weight = w;
}

public void setPrice(double cP, double mU) {
    sellPrice = addVATandMarkUp(cP, mU);
}

public void setQuantity(int quantity) {
    this.quantity = quantity;
}
```

// 1.8
// HELPER METHOD 2:

```
private String getProductTypeName(int pt) {

    String word = "";

    if (pt == TYPE_FOOD) {
        word = "Food";
    } else if (pt == TYPE_TOILETRIES) {
        word = "Toiletries";
    } else {
        word = "Other";
    }

    return word;
}
```

// 1.9
// TOSTRING METHOD:

```
@Override
public String toString() {

    String output = "";
```

```
        output = "\n" + prodID + ": " + prodName + " " + weight;

        output = output + " (" + getProductTypeName(prodType) + ")";

        output = output + "\n" + quantity + " x R" + sellPrice;

        DecimalFormat dec = new DecimalFormat("#.##");
        output = output + " = R" + Double.parseDouble(dec.format(quantity * sellPrice));

        return output;
    }

    // 1.10
    // STATIC METHOD:

    public static double getVAT() {
        return VAT;
    }

}
```

```
package productsui;

import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;

// 2.1
public class ProductsManager {

    // 2.2
    // MANAGER CLASS'S FIELDS/PROPERTIES:

    private Product item[] = new Product[150];
    private int size = 0;

    // 2.3
    // MANAGER CLASS'S CONSTRUCTOR METHOD:

    public ProductsManager(String fileName) {

        try {
            Scanner scFile = new Scanner(new File(fileName));

            while (scFile.hasNext()) {
                String line = scFile.nextLine();
```

line:

TOM01#Simba Tomato Chips#125 g#F#16.50#5#31

0 **1** **2** **3** **4** **5** **6**

4

```
            String data[] = line.split("#");

            item[size] = new Product(data[0], data[1], data[2], data[3].charAt(0),
                                    Double.parseDouble(data[4]), Double.parseDouble(data[5]),
                                    Integer.parseInt(data[6]));

            size++;
        }

    } catch (FileNotFoundException ex) {
        JOptionPane.showMessageDialog(null, "Cannot find Products.txt");
    }

}

// 2.4
// METHOD USED TO DISPLAY ALL DATA (RETURN AS ONE STRING):

public String getAllProducts() {
    String output = "";
    for (int i = 0; i < size; i++) {
        output = output + item[i].toString() + "\n";
    }
    return output;
}

// 2.5
// METHOD TO CALCULATE THE TOTAL WORTH OF ALL THE PRODUCTS IN STOCK:

public double totalWorth() {
    double total = 0;
    for (int i = 0; i < size; i++) {
        total = total + item[i].getQuantity() * item[i].getSellPrice();
    }
    return total;
}


// 2.6
// METHOD TO CALCULATE AVERAGE WEIGHT OF FOODS THAT ARE MEASURED IN G:

public double AverageWeight() {
    double total = 0;
    int count = 0;

    for (int i = 0; i < size; i++) {
        if (item[i].getProdType() == Product.TYPE_FOOD) {
            String data[] = item[i].getWeight().split(" ");          //"125 g" will be split around the
space
                                                                     // data[0]: "125"    data[1]: "g"
            double weight = Double.parseDouble(data[0]);     // weight: 125.0
            String unit = data[1];                           // unit: "g"
```

```
            if (unit.equals("g")) {
                total = total + weight;
                count++;
            }

        } //end if
    } // end for loop

    double avg = total / count;

    return avg;
}
```

// 2.7
// METHOD TO RETURN THE QUANTITY OF A CERTAIN PRODUCT

```
public int getProductQuantity(String id) {
    int qty = 0;

    for (int i = 0; i < size; i++) {

        String prodIDNumber = item[i].getProdID();

        if (prodIDNumber.equals(id)) {
            qty = item[i].getQuantity();
            i = size;                    ← //So that it will stop looping when the product was
found
        } // end if

    }  // end for

    return qty;
}
```

// 2.8
// METHOD TO DECREASE THE QUANTITY OF A PRODUCT BY A CERTAIN AMOUNT

```
public void decreaseQuantity(String id, int byAmount) {

    for (int i = 0; i < size; i++) {

        String prodIDNumber = item[i].getProdID();

        if (prodIDNumber.equals(id)) {
            int currentQty = item[i].getQuantity();
            int newQty = currentQty - byAmount;
            if (newQty >= 0) {
                item[i].setQuantity(newQty);
            }
```

```
            i = size;                    ← //So that it will stop looping when the product was
found
          } // end if

       } // end for

   }

}
```

```
package productsui;

import javax.swing.JOptionPane;

public class ProductsUI {

   public static void main(String[] args) {

      // THE FOLLOWING INSTRUCTIONS IS NOT PART OF THE PROGRAM
      // AND ARE INSERTED FOR EXPLANATION PURPOSES:

      // Creating TWO objects called product1 & product2 (instead of using arrays).
     // Displaying some of the data stored in their fields and changing others. Output:

        The product1 object contains the following data:

        TOM03: All Gold Tomato Sauce 500 ml (Food)
        18 x R56.39 = R1015.02

        15.0% VAT has been added to the purchase price of R46.70
        Yes, that is right, it has been increased by a massive 15.0%
        Further, it's cost has been marked up by 5% to R56.39

        The number of All Gold Tomato Sauce (500 ml) bottles in stock: 18
        2 bottles have been sold. There are 16 bottles left.

        The product2 object contains the following data:

        TOM04: All Gold Tomato Sauce 250 ml (Food)
        27 x R29.38 = R793.26

        15.0% VAT has been added to the purchase price of R24.80
        Yes, that is right, it has been increased by a massive 15.0%

        The number of All Gold Tomato Sauce (250 ml) bottles in stock: 27

      Product product1 = new Product("TOM03", "All Gold Tomato Sauce", "500 ml", 'F', 46.70, 5,
18);
```

```
        System.out.println("The product1 object contains the following data: \n"
                + product1.toString());                                    // OR just: + product1

        System.out.println("\n" + Product.getVAT() + "% VAT has been added to the purchase price
                                                                    of R46.70");
        System.out.println("Yes, that is right, it has been increased by a massive " + Product.VAT
                                                + "%");

        System.out.println("Further, it's cost has been marked up by 5% to R" +
product1.getSellPrice());

        int currentQty = product1.getQuantity();

        System.out.println("\nThe number of " + product1.getProdName() + " (" +
product1.getWeight()
                                        + ") bottles in stock: " + currentQty);

        product1.setQuantity(currentQty - 2);

        System.out.println("2 bottles have been sold. There are "
                                        + product1.getQuantity() + " bottles left.");

        Product product2 = new Product("TOM04", "All Gold Tomato Sauce", "250 ml", 'F', 24.80, 3,
27);

        System.out.println("\nThe product2 object contains the following data: \n"
                + product2.toString());                                    // OR just: + product2

        System.out.println("\n" + Product.getVAT() + "% VAT has been added to the purchase price
                                                                    of R24.80");

        System.out.println("Yes, that is right, it has been increased by a massive " + Product.VAT
                                                + "%");

        currentQty = product2.getQuantity();

        System.out.println("\nThe number of " + product2.getProdName() + " (" +
product2.getWeight()
                                        + ") bottles in stock: " + currentQty);

        // PLEASE NOTE that the constant VAT is static and belongs to the class and is not created
for
        // every object product1 and product2. The fields of the objects, e.g prodName, is not-static,
so
        // they will be created for every object: product1.prodName and product2.prodName
        // Now imagine you removed the word static from the declaration of VAT, then it will be
created
        // for every object: product1.VAT and product2.VAT which is unnecessary!
        // Because VAT is static there will only be one and can be used with Product.VAT

        // THE CODE OF QUESTION 3 STARTS HERE:
        // ***************************************************
```

```
        ProductsManager pm = new ProductsManager("Products.txt");

        System.out.println("\n\nALL PRODUCTS: \n" + pm.getAllProducts());

        System.out.println("\nTOTAL WORTH OF ALL THE PRODUCTS IN STOCK: R"
                                + pm.totalWorth());

        System.out.println("\n\nAVERAGE WEIGHT OF FOODS THAT ARE MEASURED IN GRAM:
"
                                + pm.AverageWeight() + " g");

        System.out.println("\n\nQUANTITY OF TOMATO CHIPS IN STOCK: "
                                + pm.getProductQuantity("TOM02"));

        System.out.println("10 PACKETS TOMATO CHIPS SOLD");

        pm.decreaseQuantity("TOM02", 10);

        System.out.println("NEW QUANTITY OF TOMATO CHIPS IN STOCK: "
                                + pm.getProductQuantity("TOM02"));

    }

}
```

See Output of the program on the next page

Output of the program:

```
ALL PRODUCTS:

TOM01: Simba Tomato Chips 125 g (Food)
31 x R19.92 = R617.52

CHE01: Fasta Pasta Cheese 65 g (Food)
186 x R15.64 = R2909.04

CAN01: B-well Canola Oil 750 ml (Food)
73 x R83.75 = R6113.75

NIV01: Nivea Body Lotion 400 ml (Toiletries)
19 x R45.21 = R858.99
```

GAR01: Smash Garlic Butter 104 g (Food)
93 x R25.36 = R2358.48

GIL01: Gillette Shaving Foam 200 ml (Toiletries)
24 x R35.04 = R840.96

TOM02: Simba Tomato Chips 30 g (Food)
85 x R9.36 = R795.6

HAN01: Handy Andy Lemon 750 ml (Other)
49 x R40.84 = R2001.16

TOTAL WORTH OF ALL THE PRODUCTS IN STOCK: R16495.5

AVERAGE WEIGHT OF FOODS THAT ARE MEASURED IN GRAM: 81.0 g

QUANTITY OF TOMATO CHIPS IN STOCK: 85
10 PACKETS TOMATO CHIPS SOLD
NEW QUANTITY OF TOMATO CHIPS IN STOCK: 75