ПРАКТИЧНА РОБОТА 9/10. ОРГАНІЗАЦІЯ ДАНИХ ТА СПРОЩЕННЯ УМОВНИХ ВИРАЗІВ

Дано програмний код класу ShoppingCart.java

```
import java.util.*;
import java.text.*;
/**
* Containing items and calculating price.
*/
public class ShoppingCart {
    public static enum ItemType { NEW, REGULAR, SECOND_FREE, SALE };

    /**
    * Tests all class methods.
    */
    public static void main(String[] args) {
    // TODO: add tests here
        ShoppingCart cart = new ShoppingCart();
        cart.addItem("Apple", 0.99, 5, ItemType.NEW);
        cart.addItem("Banana", 20.00, 4, ItemType.SECOND_FREE);
        cart.addItem("A long piece of toilet paper", 17.20, 1, ItemType.SALE);
        cart.addItem("Nails", 2.00, 500, ItemType.REGULAR);
        System.out.println(cart.formatTicket());
    }

    /**
    * Adds new item.
    *
    * @param title item title 1 to 32 symbols
    * @param price item ptice in USD, > 0
    * @param quantity item quantity, from 1
    * @param type item type
    *
    * @throws IllegalArgumentException if some value is wrong
    */
    public void addItem(String title, double price, int quantity, ItemType type){
        if (title == null || title.length() == 0 || title.length() > 32)
            throw new IllegalArgumentException("Illegal title");
        if (price < 0.01)
            throw new IllegalArgumentException("Illegal price");
        if (quantity <= 0)
            throw new IllegalArgumentException("Illegal quantity");
        Item item = new Item();
        item.title = title;
        item.price = price;
        item.quantity = quantity;
        item.type = type;
        items.add(item);
    }

    /**
    * Formats shopping price.
    *
    * @return string as lines, separated with \n,
    * first line: # Item Price Quan. Discount Total
    * second line: ------------------------------------------------------------
    * next lines: NN Title $PP.PP Q DD% $TT.TT
    * 1 Some title $.30 2 - $.60
    * 2 Some very long $100.00 1 50% $50.00
    * ...
    * 31 Item 42 $999.00 1000 - $999000.00
    * end line: ------------------------------------------------------------
```

```
* last line: 31 $999050.60
*
* if no items in cart returns "No items." string.
*/
public String formatTicket(){
    if (items.size() == 0)
        return "No items.";
    List<String[]> lines = new ArrayList<String[]>();
    String[] header = {"#","Item","Price","Quan.","Discount","Total"};
    int[] align = new int[] { 1, -1, 1, 1, 1, 1 };
    // formatting each line
    double total = 0.00;
    int index = 0;
    for (Item item : items) {
        int discount = calculateDiscount(item.type, item.quantity);
        double itemTotal = item.price * item.quantity * (100.00 - discount) / 100.00;
        lines.add(new String[]{
            String.valueOf(++index),
            item.title,
            MONEY.format(item.price),
            String.valueOf(item.quantity),
            (discount == 0) ? "-" : (String.valueOf(discount) + "%"),
            MONEY.format(itemTotal)
        });
        total += itemTotal;
    }
    String[] footer = { String.valueOf(index),"","","","",
            MONEY.format(total) };
    // formatting table
    // column max length
    int[] width = new int[]{0,0,0,0,0,0};
    for (String[] line : lines)
        for (int i = 0; i < line.length; i++)
            width[i] = (int) Math.max(width[i], line[i].length());
    for (int i = 0; i < header.length; i++)
        width[i] = (int) Math.max(width[i], header[i].length());
    for (int i = 0; i < footer.length; i++)
        width[i] = (int) Math.max(width[i], footer[i].length());
    // line length
    int lineLength = width.length - 1;
    for (int w : width)
        lineLength += w;
    StringBuilder sb = new StringBuilder();
    // header
    for (int i = 0; i < header.length; i++)
        appendFormatted(sb, header[i], align[i], width[i]);
    sb.append("\n");
    // separator
    for (int i = 0; i < lineLength; i++)
        sb.append("-");
    sb.append("\n");
    // lines
    for (String[] line : lines) {
        for (int i = 0; i < line.length; i++)
            appendFormatted(sb, line[i], align[i], width[i]);
        sb.append("\n");
    }
    if (lines.size() > 0) {
    // separator
        for (int i = 0; i < lineLength; i++)
        sb.append("-");
        sb.append("\n");
    }
    // footer
    for (int i = 0; i < footer.length; i++)
        appendFormatted(sb, footer[i], align[i], width[i]);
```

```
        return sb.toString();
    }

    // --- private section ----------------------------------------------------
    private static final NumberFormat MONEY;
    static {
        DecimalFormatSymbols symbols = new DecimalFormatSymbols();
        symbols.setDecimalSeparator('.');
        MONEY = new DecimalFormat("$#.00", symbols);
    }

    /**
    * Appends to sb formatted value.
    * Trims string if its length > width.
    * @param align -1 for align left, 0 for center and +1 for align right.
    */
     public static void appendFormatted(StringBuilder sb, String value, int align, int
width){
        if (value.length() > width)
            value = value.substring(0,width);
        int before = (align == 0)
                ? (width - value.length()) / 2
                : (align == -1) ? 0 : width - value.length();
        int after = width - value.length() - before;
        while (before-- > 0)
            sb.append(" ");
        sb.append(value);
        while (after-- > 0)
            sb.append(" ");
        sb.append(" ");
    }

    /**
    * Calculates item's discount.
    * For NEW item discount is 0%;
    * For SECOND_FREE item discount is 50% if quantity > 1
    * For SALE item discount is 70%
    * For each full 10 not NEW items item gets additional 1% discount,
    * but not more than 80% total
    */
    public static int calculateDiscount(ItemType type, int quantity){
        int discount = 0;
        switch (type) {
        case NEW:
        return 0;
        case REGULAR:
        discount = 0;
        break;
        case SECOND_FREE:
        if (quantity > 1)
        discount = 50;
        break;
        case SALE:
        discount = 70;
        break;
        }
        if (discount < 80) {
        discount += quantity / 10;
        if (discount > 80)
        discount = 80;
        }
        return discount;
    }

    /** item info */
```

```
    private static class Item {
        String title;
        double price;
        int quantity;
        ItemType type;
    }
    /** Container for added items */
    private List<Item> items = new ArrayList<Item>();
}
```

2. Хід дій

2.1 Створити шабло тестового коду JUnit для ShoppingCart.java.

```
package odz;
import org.junit.*;
import static org.junit.Assert.*;

/**
*
* @author Vycheslav
*/
public class ShoppingCartTest {

    /**
    * Test of appendFormatted method, of class ShoppingCart.
    */
    @Test
    public void testAppendFormatted() {
        System.out.println("appendFormatted");
        StringBuilder sb = null;
        String value = "";
        int align = 0;
        int width = 0;
        ShoppingCart.appendFormatted(sb, value, align, width);
        // TODO review the generated test code and remove the default call to fail.
        fail("The test case is a prototype.");
    }

    /**
    * Test of calculateDiscount method, of class ShoppingCart.
    */
    @Test
    public void testCalculateDiscount() {
        System.out.println("calculateDiscount");
        ShoppingCart.ItemType type = null;
        int quantity = 0;
        int expResult = 0;
        int result = ShoppingCart.calculateDiscount(type, quantity);
        assertEquals(expResult, result);
        // TODO review the generated test code and remove the default call to fail.
        fail("The test case is a prototype.");
    }
}
```

2.2 Модифікувати тестовий метод *testAppendFormatted()* шляхом додавання деяких assertions до тестового методу. Після цього тестовий метод має виглядати наступним чином:

```
    @Test
    public void testAppendFormatted() {
        StringBuilder sb = new StringBuilder();
        ShoppingCart.appendFormatted(sb, "SomeLine", 0, 14);
        assertEquals(sb.toString(), " SomeLine ");
        sb = new StringBuilder();
        ShoppingCart.appendFormatted(sb, "SomeLine", 0, 15);
        assertEquals(sb.toString(), " SomeLine ");
        sb = new StringBuilder();
        ShoppingCart.appendFormatted(sb, "SomeLine", 0, 5);
        assertEquals(sb.toString(), "SomeL ");
        sb = new StringBuilder();
        ShoppingCart.appendFormatted(sb, "SomeLine", 1, 15);
        assertEquals(sb.toString(), " SomeLine ");
        sb = new StringBuilder();
        ShoppingCart.appendFormatted(sb, "SomeLine", -1, 15);
        assertEquals(sb.toString(), "SomeLine ");
    }
```

2.3 Модифікувати тестовий метод *testCalculateDiscount()* шляхом додавання деяких assertions до тестового методу. Тестовий метод набуде вигляду :

```
        @Test
        public void testCalculateDiscount() {
            assertEquals(80,                                                    Shopping-
Cart.calculateDiscount(ShoppingCart.ItemType.SALE, 500));
            assertEquals(73,                                                    Shopping-
Cart.calculateDiscount(ShoppingCart.ItemType.SALE, 30));
            assertEquals(71,                                                    Shopping-
Cart.calculateDiscount(ShoppingCart.ItemType.SALE, 10));
            assertEquals(70,                                                    Shopping-
Cart.calculateDiscount(ShoppingCart.ItemType.SALE, 9));
            assertEquals(70,                                                    Shopping-
Cart.calculateDiscount(ShoppingCart.ItemType.SALE, 1));

            assertEquals(0, ShoppingCart.calculateDiscount(ShoppingCart.ItemType.NEW,
20));
            assertEquals(0, ShoppingCart.calculateDiscount(ShoppingCart.ItemType.NEW,
10));
            assertEquals(0, ShoppingCart.calculateDiscount(ShoppingCart.ItemType.NEW,
1));
            assertEquals(80,                                                    Shopping-
Cart.calculateDiscount(ShoppingCart.ItemType.SECOND_FREE, 500));
            assertEquals(53,                                                    Shopping-
Cart.calculateDiscount(ShoppingCart.ItemType.SECOND_FREE, 30));
            assertEquals(51,                                                    Shopping-
Cart.calculateDiscount(ShoppingCart.ItemType.SECOND_FREE, 10));
            assertEquals(50,                                                    Shopping-
Cart.calculateDiscount(ShoppingCart.ItemType.SECOND_FREE, 9));
            assertEquals(50,                                                    Shopping-
Cart.calculateDiscount(ShoppingCart.ItemType.SECOND_FREE, 2));
            assertEquals(0,                                                     Shopping-
Cart.calculateDiscount(ShoppingCart.ItemType.SECOND_FREE, 1));
        }
```

2.4 Виконати виділення методів *appendSeparator*, *adjustColumnWidth* та *appendFormattedLine :*

```
private appendSeparator(StringBuilder sb, int lineLength){
    for(int i = 0; i < lineLength; i++)
        sb.append("\n");
}

private void adjustColumnWidth(int[] width, String[] columns){
    for(int i = 0; i < width.length; i++)
        width[i] = (int) Math.max(width[i], columns[i].length);
}

private void appendFormattedLine(StringBuilder sb, String[] line, int[] align, int[]
width, Boolean newLine){
    for(int i = 0; i < line.length; i++)
        appendFormatted(sb, line[i], align[i], width[i]);
    if(newLine) sb.append("\n");
}
```

before

`int[] width = new int[]{0,0,0,0,0,0};`
`for (String[] line : lines)`
`  for (int i = 0; i < line.length; i++)`
`  width[i] = (int) Math.max(width[i], line[i].length());`
`for (int i = 0; i < header.length; i++)`
`  width[i] = (int) Math.max(width[i], header[i].length());`
`for (int i = 0; i < footer.length; i++)`
`  width[i] = (int) Math.max(width[i], footer[i].length());`

after

`int[] width = new int[]{0,0,0,0,0,0};`
`for (String[] line : lines)`
`  adjustColumnWidth(width, line);`
`adjustColumnWidth(width, header);`
`adjustColumnWidth(width, footer);`

before

```
        // header
        for (int i = 0; i < header.length; i++)
            appendFormatted(sb, header[i], align[i], width[i]);
        sb.append("\n");

        // separator
        for (int i = 0; i < lineLength; i++)
            sb.append("-");
        sb.append("\n");

        // lines
        for (String[] line : lines) {
            for (int i = 0; i < line.length; i++)
                appendFormatted(sb, line[i], align[i], width[i]);
            sb.append("\n");
        }
        if (lines.size() > 0) {
            // separator
            for (int i = 0; i < lineLength; i++)
                sb.append("-");
            sb.append("\n");
        }

        // footer
        for (int i = 0; i < footer.length; i++)
            appendFormatted(sb, footer[i], align[i], width[i]);
```

after

```
        // header
        appendFormatedLine(sb, header, align, width, true);
        appendSeparator(sb, lineLength);
        // lines
        for (String[] line : lines) {
            appendSeparator(sb, lineLength);
        }
        // footer
        appendFormattedLine(sb, footer, align, width, false);
```

Тестувати код.

2.5 Перемістити поля *total, discount до класу Item.* Зробити ці поля private та створити методи доступу до них

before

```
    private static class Item {
        String   title;
        double   price;
        int      quantity;
        ItemType type;
    }
```

after

```
    private static class Item {
        private String    title;
        private double    price;
        private int       quantity;
        private ItemType  type;
        private int       discount;
        private double    total;

        public String getTitle(){
            return title;
        }

        public void setTitle(String title){
            this.title = title;
        }

        public double getPrice(){
            this.price = price;
        }

        public int getQuantity(){
            this.quantity = quantity;
        }

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

        public ItemType getItemType(){
            return type;
        }

        public void setItemType(ItemType type){
            this.type = type;
        }

        public void setDiscount(int discount){
            this.discount = discount;
        }

        public int getDiscount(){
            return discount;
        }

        public double getTotalPrice(){
            return total;
        }

        public void setTotalPrice(double total){
            this.total = total;
        }
    }
```

before

```
    public void addItem(String title, double price, int quantity, ItemType type){
        if (title == null || title.length() == 0 || title.length() > 32)
            throw new IllegalArgumentException("Illegal title");
        if (price < 0.01)
            throw new IllegalArgumentException("Illegal price");
        if (quantity <= 0)
            throw new IllegalArgumentException("Illegal quantity");
        Item item = new Item();
        item.title    = title;
        item.price    = price;
        item.quantity = quantity;
        item.type     = type;
        items.add(item);
    }
```

after

```
    public void addItem(String title, double price, int quantity, ItemType type){
        if (title == null || title.length() == 0 || title.length() > 32)
            throw new IllegalArgumentException("Illegal title");
        if (price < 0.01)
            throw new IllegalArgumentException("Illegal price");
        if (quantity <= 0)
            throw new IllegalArgumentException("Illegal quantity");
        Item item = new Item();
        item.setTitle(title);
        item.setPrice(price);
        item.setQuantity(quantity);
        item.setType(type);
        items.add(item);
    }
```

before

```
        // formatting each line
        double total = 0.00;
        int    index = 0;
        for (Item item : items) {
            int discount = calculateDiscount(item.type, item.quantity);
            double itemTotal = item.price * item.quantity * (100.00 - discount) /
                                                                        100.00;
            lines.add(new String[]{
                String.valueOf(++index),
                item.title,
                MONEY.format(item.price),
                String.valueOf(item.quantity),
                (discount == 0) ? "-" : (String.valueOf(discount) + "%"),
                MONEY.format(itemTotal)
            });
            total += itemTotal;
        }
```

after

```
        // formatting each line
        double total = 0.00;
        int    index = 0;
        for (Item item : items) {
            item.setDiscount( calculateDiscount(item.getType(), item.getQuantity());
            item.setTotalPrice(item.getPrice()  *  item.getQuantity()  *  (100.00  -
item.getDiscount()) / 100.00;

            lines.add(new String[]{
                String.valueOf(++index),
                item.getTitle(),
```

```
                MONEY.format(item.getPrice()),
                String.valueOf(item.getQuantity()),
                (item.getDiscount() == 0) ? "-" : (String.valueOf(item.getDiscount())
+ "%"),
                MONEY.format(item.getTotapPrice())
            });
            total += item.getTotalPrice();
        }
```

Тестувати код.

2.6 Виконати виділення методів *calculateItemsParameters()* та *getFormattedTicketTable(double total)*. Змінені фрагменти набудуть вигляду :

before

```
    public String formatTicket(){
        if (items.size() == 0)
            return "No items.";

        List<String[]> lines = new ArrayList<String[]>();

        String[] header = {"#","Item","Price","Quan.","Discount","Total"};
        int[] align = new int[] { 1, -1, 1, 1, 1, 1 };

        // formatting each line
        double total = 0.00;
        int    index = 0;
        for (Item item : items) {
            int discount = calculateDiscount(item.type, item.quantity);
            double itemTotal = item.price * item.quantity * (100.00 - discount) /
                                                                        100.00;
            lines.add(new String[]{
                String.valueOf(++index),
                item.title,
                MONEY.format(item.price),
                String.valueOf(item.quantity),
                (discount == 0) ? "-" : (String.valueOf(discount) + "%"),
                MONEY.format(itemTotal)
            });
            total += itemTotal;
        }

        String[] footer = { String.valueOf(index),"","","","",
            MONEY.format(total) };

        // formatting table

        // column max length
        int[] width = new int[]{0,0,0,0,0,0};
        for (String[] line : lines)
            adjustColumnWidth(width, line);
        adjustColumnWidth(width, header);
        adjustColumnWidth(width, footer);

        // line length
        int lineLength = width.length - 1;
        for (int w : width)
            lineLength += w;

        StringBuilder sb = new StringBuilder();

        // header
```

```
        appendFormattedLine(sb, header, align, width, true);
        // separator
        appendSeparator(sb, lineLength);
        // lines
        for (String[] line : lines) {
            appendSeparator(sb, lineLength);
        }
        // footer
        appendFormattedLine(sb, footer, align, width, false);
        return sb.toString();
    }
```

after

```
    public String formatTicket(){
        double total = calculateItemsParameters();
        return getFormattedTicketTable(total);
    }

    private double calculateItemsParameters(){
        double total = 0.00;
        int    index = 0;
        for (Item item : items) {
            int discount = calculateDiscount(item.getType(), item.getQuantity());
            item.setTotalPrice(item.getPrice()  *  item.getQuantity()  *  (100.00  -
item.getDiscount())/ 100.00;
            total += item.getTotalPrice();
        }
        return total;
    }

    private String getFormattedTicketTable(double total){
        if (items.size() == 0)
        return "No items.";
        List<String[]> lines = new ArrayList<String[]>();

        String[] header = {"#","Item","Price","Quan.","Discount","Total"};
        int[] align = new int[] { 1, -1, 1, 1, 1, 1 };

        // formatting each line
        double total = 0.00;
        int    index = 0;
        for (Item item : items) {
            int discount = calculateDiscount(item.getType(), item.getQuantity());
            item.setTotalPrice(item.getPrice() * item.getQuantity() * (100.00 -
 item.getDiscount())/100.00;
            lines.add(new String[]{
                String.valueOf(++index),
                item.getTitle(),
                MONEY.format(item.getPrice()),
                String.valueOf(item.getQuantity()),
                (item.getDiscount() == 0) ? "-" : (String.valueOf(item.getDiscount())
+ "%"),
                MONEY.format(item.getTotalPrice())
            });
            total += itemTotal;
        }

        String[] footer = { String.valueOf(index),"","","","",
            MONEY.format(total) };
        // column max length
        int[] width = new int[]{0,0,0,0,0,0};
        for (String[] line : lines)
            adjustColumnWidth(width, line);
        adjustColumnWidth(width, header);
```

```
        adjustColumnWidth(width, footer);

        // line length
        int lineLength = width.length - 1;
        for (int w : width)
            lineLength += w;

        StringBuilder sb = new StringBuilder();

        // header
        appendFormattedLine(sb, header, align, width, true);
        // separator
        appendSeparator(sb, lineLength);
        // lines
        for (String[] line : lines) {
            appendSeparator(sb, lineLength);
        }
        if(lines.size() > 0){
            appendSeparator(sb, lineLength);
        }
        // footer
        appendFormattedLine(sb, footer, align, width, false);
        return sb.toString();
    }
```

Виконати тестування коду.

2.7 Виконати виділення методу *convertItemsToTableLines()*

```
    private List<String[]> convertItemsToTableLines(){
       List<String[]> lines = new ArrayList<String[]>();
       int index = 0;
       for (Item item : items) {
           lines.add(new String[]{
               String.valueOf(++index),
               item.getTitle(),
               MONEY.format(item.getPrice()),
               String.valueOf(item.getQuantity()),
               (item.getDiscount() == 0) ? "-" : (String.valueOf(item.getDiscount())
+ "%"),
               MONEY.format(item.getTotalPrice())
           });
       }
       return lines;
    }
```

before

```
        List<String[]> lines = new ArrayList<String[]>();

        String[] header = {"#","Item","Price","Quan.","Discount","Total"};
        int[] align = new int[] { 1, -1, 1, 1, 1, 1 };

        // formatting each line
        double total = 0.00;
        int    index = 0;
        for (Item item : items) {
            int discount = calculateDiscount(item.getType(), item.getQuantity());
            double itemTotal = item.getPrice() * item.getQuantity() * (100.00 -
item.getDiscount()) / 100.00;
            lines.add(new String[]{
                String.valueOf(++index),
                item.getTitle(),
```

```
                MONEY.format(item.getPrice()),
                String.valueOf(item.getQuantity()),
                (item.getDiscount() == 0) ? "-" : (String.valueOf(item.getDiscount())
+ "%"),
                MONEY.format(item.getTotalPrice())
            });
            total += itemTotal;
        }

        String[] footer = { String.valueOf(index),"","","","",
            MONEY.format(total) };
```

after

```
    String[] header = {"#","Item","Price","Quan.","Discount","Total"};
    int[] align = new int[] { 1, -1, 1, 1, 1, 1 };
    List<String[]> lines = convertItemsToTableLines();
    String[] footer = { String.valueOf(index),"","","","",
            MONEY.format(total) };
```

Тестувати код.

2.8 Знайти інші недоліки коду і виконати їх рефакторинг.