

Міністерство освіти і науки України
Національний університет «Львівська політехніка»
Кафедра «Електронних обчислювальних машин»



Звіт
з лабораторної роботи № 2
з дисципліни: «Кросплатформенні засоби програмування»
на тему: «Дослідження базових конструкцій мови Java»

Виконав:

студент групи KI-306

Олесько Б. А.

Прийняв:

доцент кафедри ЕОМ

Іванов Ю. С.

Мета: ознайомитися з процесом розробки класів та пакетів мовою Java.

Завдання (варіант № 11)

11. Монітор

Написати та налагодити програму на мові Java, що реалізує у вигляді класу предметну область згідно варіанту. Програма має задовольняти наступним вимогам:

- програма має розміщуватися в пакеті Група.Прізвище.Lab2;
- клас має містити мінімум 3 поля, що є об'єктами класів, які описують складові частини предметної області;
- клас має містити кілька конструкторів та мінімум 10 методів;
- для тестування і демонстрації роботи розробленого класу розробити клас-драйвер;
- методи класу мають вести протокол своєї діяльності, що записується у файл;
- розробити механізм коректного завершення роботи з файлом (не надіятися на метод `finalize()`);
- програма має володіти коментарями, які дозволять автоматично згенерувати документацію до розробленого пакету.

2. Автоматично згенерувати документацію до розробленої програми.

3. Завантажити код на GitHub згідно методичних вказівок по роботі з GitHub.

4. Скласти звіт про виконану роботу з приведенням тексту програми, результату її виконання та фрагменту згенерованої документації та завантажити його у ВНС.

5. Дати відповідь на контрольні запитання. 4. Дати відповіді на контрольні запитання:

1. Синтаксис визначення класу.
2. Синтаксис визначення методу.
3. Синтаксис оголошення поля.
4. Як оголосити та ініціалізувати константне поле?

5. Які є способи ініціалізації полів?
6. Синтаксис визначення конструктора.
7. Синтаксис оголошення пакету.
8. Як підключити до програми класи, що визначені в зовнішніх пакетах?
9. В чому суть статичного імпорту пакетів?
10. Які вимоги ставляться до файлів і каталогів при використанні пакетів?

Вихідний код програми

Файл monitor.java

```
package KI306.Olesko.Lab3;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.PrintWriter;
/**
 * This class represents a monitor with various attributes and functionality.
 */
public class monitor {
    private boolean additional;
    private String additional_devices;
    private String type;
    private int price;
    private int count;
    private String version;
    private boolean isavailable;
    /**
     * Default constructor initializes an object with default values.
     */
    public monitor() {
        additional = true;
        additional_devices = "mouse";
        type = "AAAd27O";
        version = "0.1";
        count = 1;
        price = 100;
        isavailable = true;
    }
    /**
     * Constructor with parameters initializes an object with specified values.
     *
     * @param type          The type of the monitor.
     * @param version       The version of the monitor.
     * @param price         The price of the monitor.
     * @param count         The quantity of available monitors.
     * @param isavailable   The availability status of the monitor.
     * @param additional    Whether there are additional devices for the monitor.
     * @param additional_devices The additional devices for the monitor.
     */
    public monitor(String type, String version, int price, int count, boolean isavailable, boolean additional, String additional_devices) {
        this.additional = additional;
        this.additional_devices = additional_devices;
        this.type = type;
        this.version = version;
        this.price = price;
        this.count = count;
        this.isavailable = isavailable;
    }
    /**
     * Checks if the monitor has additional devices and logs the result.
     */
    public void checkAdditional() {
        if (!additional) {
            writeToLogFile("This monitor does not have additional devices.");
        }
        writeToLogFile("This monitor has additional devices.");
    }
}
```

```

    }
    /**
    * Changes the additional devices for the monitor and logs the change.
    *
    * @param devices The new additional devices.
    */
    public void changeAdditionalDevices(String devices) {
        if (!additional) {
            writeToLogFile("This monitor does not have additional devices.");
        }
        writeToLogFile("You changed additional devices from " + additional_devices + " to " + devices);
        this.additional_devices = devices;
    }
    /**
    * Simulates a monitor purchase and logs the result.
    *
    * @param number The number of monitors to purchase.
    */
    public void buy(int number) {
        if (count < number) {
            writeToLogFile("Oops! You want to buy more than we have available.");
        }
        this.count += number;
        writeToLogFile("You bought " + number + " monitors. Current count: " + count);
    }
    /**
    * Turns off the availability of the monitor and logs the status change.
    */
    public void turnOffAvailable() {
        this.isavailable = false;
        writeToLogFile("Monitor is now unavailable.");
    }
    /**
    * Changes the type of the monitor and logs the change.
    *
    * @param newType The new type for the monitor.
    */
    public void changeType(String newType) {
        this.type = newType;
        writeToLogFile("Type changed to: " + type);
    }
    /**
    * Changes the version of the monitor and logs the change.
    *
    * @param newVersion The new version for the monitor.
    */
    public void changeVersion(String newVersion) {
        this.version = newVersion;
        writeToLogFile("Version changed to: " + version);
    }
    /**
    * Logs the current status of the monitor.
    */
    public void status() {
        writeToLogFile("Type: " + type + "\nVersion: " + version + "\nPrice: " + price + "\nCount: " + count + "\nAvailability: " +
isavailable);
    }
    /**
    * Adds a specified amount to the price of the monitor and logs the new price.
    *
    * @param prices The amount to add to the price.
    */
    public void addPrice(int prices) {
        price += prices;
        writeToLogFile("New price: " + price);
    }
    /**
    * Adds a specified amount to the count of available monitors and logs the new count.
    *
    * @param counts The amount to add to the count.
    */
    public void addCount(int counts) {
        count += counts;
        writeToLogFile("New count: " + count);
    }
    /**

```

```

    * Clears the log file.
    */
    public void clearLogFile() {
        File logFile = new File("Olesko.txt");
        try {
            PrintWriter writer = new PrintWriter(logFile);
            writer.close();
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        }
    }
    /**
     * Writes a message to the log file.
     *
     * @param message The message to be written to the log file.
     */
    private void writeToLogFile(String message) {
        try (PrintWriter writer = new PrintWriter(new FileOutputStream(new File("Olesko.txt"), true))) {
            writer.println(message);
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        }
    }
}

```

Файл monitorapp.java

```

package KI306.Olesko.Lab3;

public class monitorapp {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        monitor monitor = new monitor();
        monitor.clearLogFile();
        monitor.status();
        monitor.addCount(5);
        monitor.addPrice(200);
        monitor.buy(3);
        monitor.changeType("AAADC47");
        monitor.changeVersion("DDF212");
        monitor.checkAdditional();
        monitor.changeAdditionalDevices("loudspeakers");
        monitor.turnOffAvailable();
    }
}

```

```

Type: AAd270
Version: 0.1
Price: 100
Count: 1
Availability: true
New count: 6
New price: 300
You bought 3 monitors. Current count: 3
Type changed to: AAADC47
Version changed to: DDF212
This monitor has additional devices.
You changed additional devices from mouse to loudspeakers
Monitor is now unavailable.

```

Module `java.base`
Package `java.lang`
Class `Object`
`java.lang.Object`

`public class Object`

Class `Object` is the root of the class hierarchy. Every class has `Object` as a superclass. All objects, including arrays, implement the methods of this class.

Since:

1.0

See Also:

Class

Constructor Summary

Constructors

Constructor	Description
<code>Object ()</code>	Constructs a new object.

Method Summary

All Methods

Instance Methods

Concrete Methods

Deprecated Methods

Modifier and Type	Method	Description
<code>protected Object</code>	<code>clone()</code>	Creates and returns a copy of this object.

Package `KI306.Olesko.Lab3`

package `KI306.Olesko.Lab3`

Classes

Class	Description
<code>monitor</code>	This class represents a monitor with various attributes and functionality.

Відповіді на контрольні запитання

1- `[public] class НазваКласу`

{

`[конструктори]`

`[методи]`

`[поля]`

}

2-`[СпецифікаторДоступу] [static] [final] Тип назваМетоду([параметри]) [throws класи]`

{

`[Тіло методу]`

`[return [значення]];`

}

3- `[СпецифікаторДоступу] [static] [final] Тип НазваПоля [= ПочатковеЗначення];`

`private int i;`

4-за допомогою ключового слова `final`

`public final int myConstant = 10;`

5- Ініціалізацію полів може здійснюватися:

У конструкторі;

Явно при оголошенні поля;

У блоці ініціалізації (виконується перед виконанням конструктора).

6- `[СпецифікаторДоступу] НазваКласу([параметри])`

```
{  
Тіло конструктора  
}  
7- package НазваПакету{.НазваПідпакету};
```

8-Тре імпортувати пакет з назвою класу:

```
import package_name.ClassName;
```

9-Суть статичного імпорту пакетів полягає в тому що непотрібно вказувати повну кваліфіковану назву цих класів або методів кожного разу, коли ви використовуєте їх у своєму коді.

10-Головною вимогою є запобігання конфлікту імен.

Висновок

Я навчився основам програмування класів на java і роботи з полями конструкторами, оголошення пакету.