Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»
Інститут комп'ютерних технологій, автоматики та метрології

Кафедра ЕОМ

Звіт
До лабораторної роботи №3
З дисципліни: «Кросплатформні засоби програмування»
На тему «Основи розробки програм мовою Java»
Варіант №16

Виконав: ст. гр. КІ-36
Струтинський В.О.
Прийняв:
Іванов Ю.С.

Львів – 2022

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

ЗАВДАННЯ

1. 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.Lab3`;
   - 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 розробленого пакету.
3. Скласти звіт про виконану роботу з приведенням тексту програми, результату її виконання та фрагменту згенерованої документації.
4. Дати відповідь на контрольні запитання.

16. Аудіоплеєр

Хід роботи:

Лістинг прогами:

```java
/**
 * Class Hard Disk
 * @author
 * @version
 */
public class HardDisk
{
    private double capacity;
    private String producer;

    /**
     * Constructor
     * @param capacity
     * @param producer
     */
    public HardDisk(double capacity, String producer)
    {
        this.capacity = capacity;
        this.producer = producer;
    }

    /**
     * Getter for capacity
     * @return capacity
     */
    public double getCapacity() {
        return capacity;
    }
```

```java
    /**
     * Setter for capacity
     * @param capacity
     */
    public void setCapacity(double capacity) {
        this.capacity = capacity;
    }

    /**
     * Getter for producer
     * @return producer
     */
    public String getProducer() {
        return producer;
    }

    /**
     * Setter for producer
     * @param producer
     */
    public void setProducer(String producer) {
        this.producer = producer;
    }

    @Override
    public String toString() {
        return "HardDisk{ " +
                "capacity = " + capacity + " mb." +
                ", producer = '" + producer + '\'' +
                '}';
    }
}

/**
 * Class Screen
 * @author
 * @version 1.0
 */
public class Screen
{
    private double diagonal;
    private String expansion;

    /**
     * Constructor
     * @param diagonal
     * @param expansion
     */
    public Screen(double diagonal, String expansion) {
        this.diagonal = diagonal;
        this.expansion = expansion;
    }

    /**
     * Getter for Diagonal
     * @return diagonal
     */
    public double getDiagonal() {
        return diagonal;
    }

    /**
     * Setter for diagonal
```

```
     * @param diagonal
     */
    public void setDiagonal(double diagonal) {
        this.diagonal = diagonal;
    }

    /**
     * Getter for expansion
     * @return
     */
    public String getExpansion() {
        return expansion;
    }

    /**
     * Setter for expansion
     * @param expansion
     */
    public void setExpansion(String expansion) {
        this.expansion = expansion;
    }

    @Override
    public String toString() {
        return "Screen{ " +
                "diagonal = " + diagonal +
                ", expansion = '" + expansion + '\'' +
                '}';
    }
}
import java.io.*;
import java.text.SimpleDateFormat;
import java.util.*;

/**
 * Class Logger. Was created to log information, errors and warnings. Also
there was implemented Singelton
 * @author
 * @version 1.0
 */
public class Logger
{
    private static Logger logger;
    private final String fileName;

    protected final String infoFlag = new String("[INFO] ");
    protected final String errorFlag = new String("[ERROR] ");
    protected final String warningFlag = new String("[WARNING] ");

    /**
     * Constructor
     * @param fileName
     */
    private Logger(String fileName)
    {
        this.fileName = fileName;
        File loggerFile = null;
        FileWriter fout = null;
        try
        {
            loggerFile = new File(fileName);
            fout = new FileWriter(loggerFile, true);
            SimpleDateFormat formatter= new SimpleDateFormat("yyyy-MM-dd 'at'
```

```
HH:mm:ss z");
            Date date = new Date(System.currentTimeMillis());
            fout.write("[" + formatter.format(date) + "] " + "Logger start to
work\n");
        }
        catch (IOException e)
        {
            System.err.println("Something wrong with log file" +
e.getMessage());
            System.exit(1);
        }
        finally
        {
            try
            {
                fout.flush();
                fout.close();
            }
            catch (IOException e)
            {
                System.out.println(e.getMessage());
            }
        }
    }

    /**
     * Method to do logging
     * @param massege
     */
    public void log(String massege)
    {
        File loggerFile = null;
        FileWriter fout = null;
        try
        {
            loggerFile = new File(this.fileName);
            fout = new FileWriter(loggerFile, true);
            SimpleDateFormat formatter= new SimpleDateFormat("yyyy-MM-dd 'at'
HH:mm:ss z");
            Date date = new Date(System.currentTimeMillis());
            fout.write("[" + formatter.format(date) + "] " + massege +  "\n");
        }
        catch (IOException e)
        {
            System.err.println("Something wrong with log file" +
e.getMessage());
            System.exit(1);
        }
        finally
        {
            try
            {
                fout.flush();
                fout.close();
            }
            catch (IOException | NullPointerException e)
            {
                System.out.println(e.getMessage());
            }
        }
    }

    /**
```

```java
     * Singleton implementation
     * @param fileName
     * @return
     */
    public static Logger getLogger(String fileName)
    {
        if (logger == null)
        {
            logger = new Logger(fileName);
        }
        return logger;
    }

    /**
     * Getter for logger
     * @return logger
     */
    public static Logger getLogger()
    {
        return logger;
    }

}
/**
 * Class Button
 * @author
 * @version 1.0
 */
public class Button
{
    private String action;

    /**
     * Constructor
     * @param action
     */
    public Button(String action) {
        this.action = action;
    }

    /**
     * Getter for action
     * @return action
     */
    public String getAction() {
        return action;
    }

    /**
     * Setter for action
     * @param action
     */
    public void setAction(String action) {
        this.action = action;
    }

    @Override
    public String toString() {
        return "Button{ " +
                "action = '" + action + '\'' +
                '}';
    }
}
```

```java
import org.w3c.dom.ls.LSOutput;

import javax.sound.midi.Soundbank;
import java.util.ArrayList;

/**
 * Class AudioPlayer
 * @author
 * @version 1.0
 */
public class AudioPlayer
{
    private final Button nextSong = new Button("next song");
    private final Button prevSong = new Button("prev song");
    private final Button pause = new Button("pause");
    private Logger logger = Logger.getLogger("logs.txt");
    private Screen screen;
    private HardDisk hardDisk;
    private ArrayList<String> songs = new ArrayList<>();
    private int curSong = 0;

    /**
     * Constructor
     * @param screen
     * @param hardDisk
     */
    public AudioPlayer(Screen screen, HardDisk hardDisk) {
        logger.log(logger.infoFlag + "AudioPlayer constructor called");
        this.screen = screen;
        this.hardDisk = hardDisk;
    }

    /**
     * Method to add new song to player
     * @param song
     */
    public void AddSong(String song)
    {
        songs.add(song);
        System.out.println(song + " was added to audio player");
        logger.log(logger.infoFlag + "AudioPlayer AddSong method was called");
    }


    /**
     * Method to turn on next song
     */
    public void TurnOnNextSong()
    {
        logger.log(logger.infoFlag + "TurnOnNextSong AudioPlayer method was
called");
        if(curSong == songs.size() - 1)
        {
            System.out.println("You push button " + nextSong.getAction());
            System.out.println("Now playing " + songs.get(curSong));
            curSong = 0;
        } else if (curSong < songs.size() - 1) {
            System.out.println("You push button " + nextSong.getAction());
            System.out.println("Now playing " + songs.get(curSong));
            curSong++;
        }
    }
```

```
    /**
     * Method to turn on prev song
     */
    public void TurnOnPrevSong()
    {
        logger.log(logger.infoFlag + "TurnPrevNextSong AudioPlayer method was
called");
        if(curSong == 0)
        {
            System.out.println("You push button " + prevSong.getAction());
            System.out.println("Now playing " + songs.get(curSong));
            curSong = songs.size() - 1;
        } else if (curSong > 0) {
            System.out.println("You push button " + prevSong.getAction());
            System.out.println("Now playing " + songs.get(curSong));
            curSong--;
        }
    }

    public Button getNextSong() {
        return nextSong;
    }

    public Button getPrevSong() {
        return prevSong;
    }

    public Button getPause() {
        return pause;
    }

    public Logger getLogger() {
        return logger;
    }

    public void setLogger(Logger logger) {
        this.logger = logger;
    }

    public Screen getScreen() {
        return screen;
    }

    public void setScreen(Screen screen) {
        this.screen = screen;
    }

    public HardDisk getHardDisk() {
        return hardDisk;
    }

    public void setHardDisk(HardDisk hardDisk) {
        this.hardDisk = hardDisk;
    }

    public ArrayList<String> getSongs() {
        return songs;
    }

    public void setSongs(ArrayList<String> songs) {
        this.songs = songs;
    }
```

```
    public int getCurSong() {
        return curSong;
    }

    public void setCurSong(int curSong) {
        this.curSong = curSong;
    }

    @Override
    public String toString() {
        return "AudioPlayer{ " +
                " screen=" + screen + "\n" +
                ", hardDisk=" + hardDisk +  "\n" +
                ", songs=" + songs + "\n" +
                ", curSong=" + curSong + "\n" +
                '}';
    }
}
```

Результат:

```
Stepan Giga - Zoloto Karpat was added to audio player
Stepan Giga - Yvoruna was added to audio player
Victor Pavlic - Shikidim was added to audio player
Zhadan i Sobaku - Madona was added to audio player
Zhadan i Sobaku - Kobzon was added to audio player
You push button prev song
Now playing Stepan Giga - Zoloto Karpat
You push button prev song
Now playing Zhadan i Sobaku - Kobzon
You push button prev song
Now playing Zhadan i Sobaku - Madona
You push button prev song
Now playing Victor Pavlic - Shikidim
You push button prev song
Now playing Stepan Giga - Yvoruna
You push button prev song
Now playing Stepan Giga - Zoloto Karpat
You push button prev song
Now playing Zhadan i Sobaku - Kobzon
AudioPlayer{  screen=Screen{ diagonal = 7.8, expansion = '720x1980'}
, hardDisk=HardDisk{ capacity = 1000.0 mb., producer = 'Harman'}
, songs=[Stepan Giga - Zoloto Karpat, Stepan Giga - Yvoruna, Victor Pavlic - Shikidim, Zhadan i Sobaku - Madona, Zhadan i Sobaku - Kobzon]
, curSong=3
}

Process finished with exit code 0
```

Висновок: у ході данної лабораторної роботи я ознайомився з процесом розробки класів та пакетів мовою Java.