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 «Спадкування та інтерфейси»

# Варіант №16

Виконав: ст. гр. КІ-305

Лавро В. В.

Прийняв:

Іванов Ю. С.

**Львів – 2023**

**Мета:** ознайомитися з спадкуванням та інтерфейсами у мові Java.

**ЗАВДАННЯ**

1. Написати та налагодити програму на мові Java, що розширює клас, що реалізований у лабораторній роботі №3, для реалізації предметної області заданої варіантом. Суперклас, що реалізований у лабораторній роботі №3, зробити абстрактним. Розроблений підклас має забезпечувати механізми свого коректного функціонування та реалізовувати мінімум один інтерфейс. Програма має розміщуватися в пакеті **Група.Прізвище.Lab3** та володіти коментарями, які дозволять автоматично згенерувати документацію до розробленого пакету.
2. Автоматично згенерувати документацію до розробленого пакету.
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.



**Виконання:**

**Лістинг програми:**

abstract class AudioPlayer

import java.util.ArrayList;  
  
*/\*\*  
 \* Class AudioPlayer  
 \* @author  
 \* @version 1.0  
 \*/*public abstract class AudioPlayer  
{  
 protected final Button nextSong = new Button("next song");  
 protected final Button prevSong = new Button("prev song");  
 protected final Button pause = new Button("pause");  
 protected Logger logger = Logger.*getLogger*("logs.txt");  
 protected Screen screen;  
 protected HardDisk hardDisk;  
 protected ArrayList<String> songs = new ArrayList<>();  
 protected 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 abstract void TurnOnNextSong();  
  
 */\*\*  
 \* Method to turn on prev song  
 \*/* public abstract void TurnOnPrevSong();  
  
  
 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" +  
 '}';  
 }  
}

class Button

*/\*\*  
 \* 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 + '\'' +  
 '}';  
 }  
}

class Dictaphone

*/\*\*  
 \* Class Dictaphone  
 \* @version 1.0  
 \*/*public class Dictaphone extends AudioPlayer implements VoiceRecord  
{  
 */\*\*  
 \* Constructor  
 \*  
 \* @param screen  
 \* @param hardDisk  
 \*/* public Dictaphone(Screen screen, HardDisk hardDisk) {  
 super(screen, hardDisk);  
 }  
  
 */\*\*  
 \* Overrided method to turn on next song  
 \*/* @Override  
 public void TurnOnNextSong() {  
 logger.log(logger.infoFlag + "TurnOnNextSong Dictaphone 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++;  
 }  
 }  
 */\*\*  
 \* Overrided method to turn on prev song  
 \*/* @Override  
 public void TurnOnPrevSong() {  
 logger.log(logger.infoFlag + "TurnPrevNextSong Dictaphone 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--;  
 }  
 }  
  
 @Override  
 public void RecordVoice(String voice) {  
 songs.add("Voice Record - " + voice);  
 }  
  
 @Override  
 public String toString() {  
 return "Dictaphone{ " +  
 ", screen=" + screen + "\n" +  
 ", hardDisk=" + hardDisk + "\n" +  
 ", songs=" + songs + "\n" +  
 ", curSong=" + curSong + "\n" +  
 '}';  
 }  
}

class HardDisk

*/\*\*  
 \* 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;  
 }  
  
 */\*\*  
 \* 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 Logger

import java.io.File;  
import java.io.FileWriter;  
import java.io.IOException;  
import java.text.SimpleDateFormat;  
import java.util.Date;  
  
*/\*\*  
 \* 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());  
 }  
 }  
 }  
  
 */\*\*  
 \* 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 Screen

*/\*\*  
 \* 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 + '\'' +  
 '}';  
 }  
}

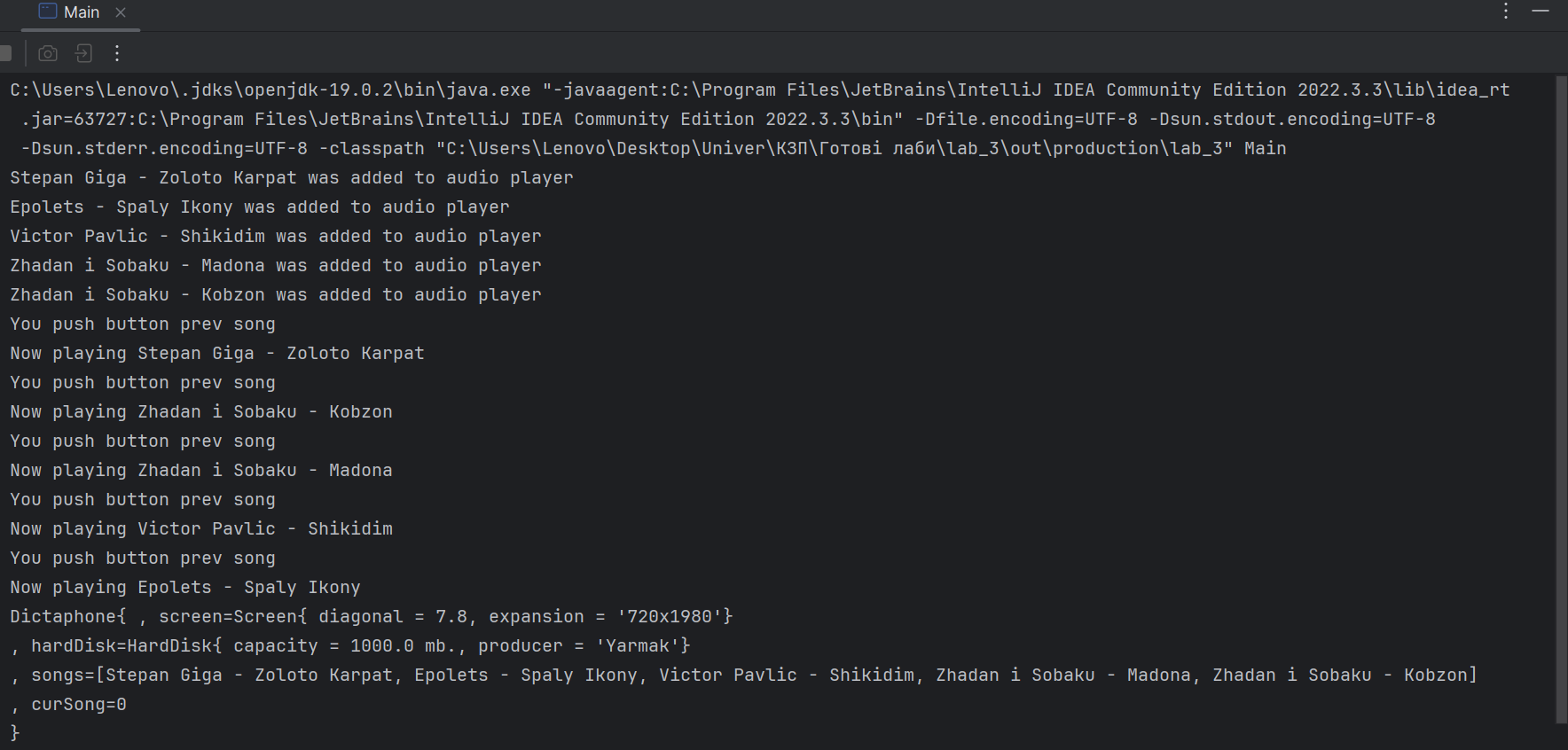
interface VoiceRecords

public interface VoiceRecord  
{  
 void RecordVoice(String voice);  
}

class Main

public class Main{  
 public static void main(String[] args) {  
 Dictaphone Dictaphone = new Dictaphone(new Screen(7.8, "720x1980"), new HardDisk(1000, "Yarmak"));  
  
 Dictaphone.AddSong("Stepan Giga - Zoloto Karpat");  
 Dictaphone.AddSong("Epolets - Spaly Ikony");  
 Dictaphone.AddSong("Victor Pavlic - Shikidim");  
 Dictaphone.AddSong("Zhadan i Sobaku - Madona");  
 Dictaphone.AddSong("Zhadan i Sobaku - Kobzon");  
  
 Dictaphone.TurnOnPrevSong();  
 Dictaphone.TurnOnPrevSong();  
  
 Dictaphone.TurnOnPrevSong();  
 Dictaphone.TurnOnPrevSong();  
 Dictaphone.TurnOnPrevSong();  
  
 Dictaphone.RecordVoice("Hello, my name is Java");  
 System.*out*.println(Dictaphone);  
 }  
}

**Результати:**

****

**Висновок:** ознайомився з спадкуванням та інтерфейсами у мові Java.