Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет

информатики и радиоэлектроники»

Кафедра инженерной психологии и эргономики

­­­­­

Современные языки программирования

Отчет по лабораторной работе №1

«Использование языка программирования Kotlin»

Выполнил: Усов А.М.

Студент группы 310901

Преподаватель: Усенко Ф. В.

Минск 2024

Цель: Выполнить разработку приложения с использованием языка программирования Kotlin.

Задание.Вариант 30.

Добавить класс Плейлист, хранящий список видео. Класс должен выдавать следующее видео из списка при вызове метода Воспроизвести плейлист(). Добавить класс Музыкальное видео.

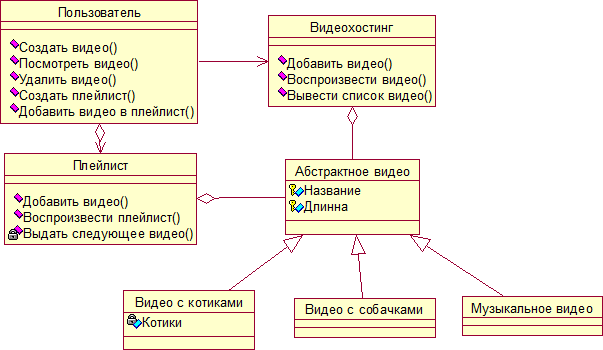


Рисунок 60 – Диаграмма классов

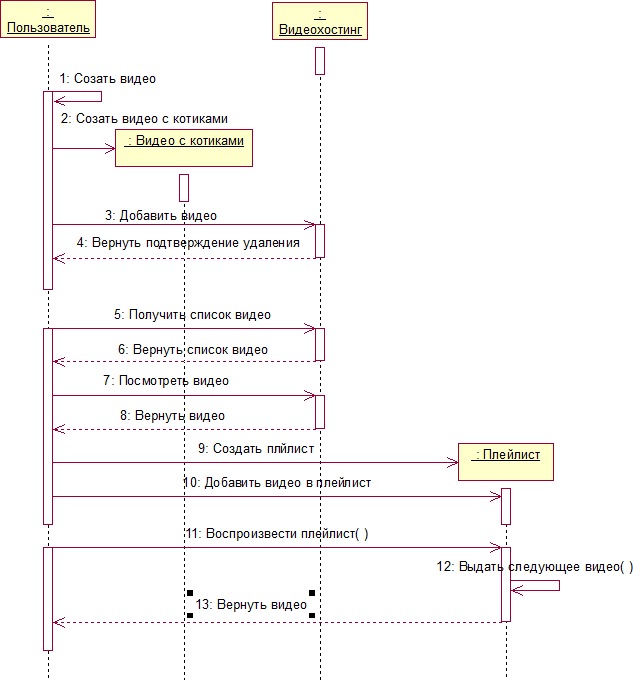


Рисунок 61 – Диаграмма последовательности

Листинг кода:

Файл Main.kt:

import client.Client  
import hosting.VideoHosting  
import videos.\*  
  
  
fun main(args: Array<String>){  
 val client = Client()  
 while (true){  
 *println*()  
 *println*("========================")  
 *println*()  
 *println*("1. Create video")  
 *println*("2. Get video list")  
 *println*("3. Delete video")  
 *println*("4. Watch video")  
 *println*("5. Create playlist")  
 *println*("6. Add video to playlist")  
 *println*("7. Start playlist")  
 *println*("8. Exit");  
  
 var choice:Number;  
 try{  
 choice = *readln*().*toInt*()  
 }catch (e:Exception){  
 *println*("invalid Input")  
 continue;  
 }  
  
  
  
 when(choice){  
 1->{  
 client.createVideo();  
 }  
 2->{  
 var videos = client.getVideoList();  
 if(videos.size == 0){  
 *println*("No videos found")  
 }else{  
 *println*("Videos:")  
 videos.*forEach* **{** *println*(**it**.Name + " "+**it**.Duration)  
 **}** }  
 }  
 3->{  
 *println*("Enter video name: ");  
 var name = *readln*();  
 client.deleteVideo(name);  
 }  
 4->{  
 *println*("Enter video name: ");  
 var name = *readln*();  
 client.watchVideo(name);  
 }  
 5->{  
 client.createPlaylist();  
 }  
 6->{  
 client.addVideoToPlayList();  
 }  
 7->{  
 *println*("Enter playlist name: ");  
 var name = *readln*();  
 client.startPlaylist(name);  
 }  
 8 ->{  
 break;  
 }  
 }  
  
 }  
  
  
}

Файл Client.kt:

package client  
import hosting.VideoHosting  
import videos.\*;  
  
abstract class ClientBase {  
 abstract fun createVideo();  
 abstract fun getVideoList():List<Video>;  
 abstract fun watchVideo(name: String);  
 abstract fun deleteVideo(name: String);  
 abstract fun createPlaylist();  
 abstract fun addVideoToPlayList();  
  
 abstract fun startPlaylist(Name:String);  
}  
  
  
  
class Client:ClientBase(){  
  
  
 private val hosting = VideoHosting();  
 private val playlists = *mutableListOf*<playList>();  
  
  
 init {  
 hosting.addVideo(Cat\_Video("1",10))  
 hosting.addVideo(Cat\_Video("Cat Video 2",10))  
 hosting.addVideo(Cat\_Video("Cat Video 3",10))  
 hosting.addVideo(Dog\_Video("Dog Video 1",10))  
 hosting.addVideo(Dog\_Video("Dog Video 2",10))  
 hosting.addVideo(Dog\_Video("Dog Video 3",10))  
 hosting.addVideo(Music\_Video("Music Video 1",10))  
 hosting.addVideo(Music\_Video("Music Video 2",10))  
 hosting.addVideo(Music\_Video("Music Video 3",10))  
 playlists.add(playList("1"))  
 playlists[0].addVideo(hosting.getVideo("1")!!)  
 playlists[0].addVideo(hosting.getVideo("Cat Video 2")!!)  
  
 }  
  
 override fun createVideo(){  
 *println*("Enter video name: ");  
 var name = *readLine*()!!;  
 *println*("Enter video duration: ");  
 var duration:Number;  
 try{  
 duration= *readLine*()!!.*toInt*();  
 }catch (e:Exception){  
 *println*("InvalidInput");  
 return;  
 }  
 *println*("Enter video type: ");  
 *println*("1. Cat Video");  
 *println*("2. Dog Video");  
 *println*("3. Music Video");  
 var type:Number;  
 try{  
 type = *readLine*()!!.*toInt*();  
 }catch(e:Exception){  
 *println*("Invalid Input");  
 return;  
 }  
 if(type>3 || type<1){  
 *println*("Invalid type");  
 return;  
 }  
 when(type){  
 1->{  
 var video = Cat\_Video(name,duration);  
 hosting.addVideo(video);  
 }  
 2->{  
 var video = Dog\_Video(name,duration);  
 hosting.addVideo(video);  
 }  
 3->{  
 var video = Music\_Video(name,duration);  
 hosting.addVideo(video);  
 }  
 }  
 }  
  
 override fun watchVideo(name: String){  
 var video = hosting.getVideo(name);  
 if(video == null){  
 *println*("Video not found");  
 return;  
 }  
 video.Vatch();  
 }  
  
 override fun deleteVideo(name: String){  
 var video = hosting.getVideo(name);  
 if(video == null){  
 *println*("Video not found");  
 return;  
 }  
 *println*("Deleting video: ${video.Name}. Are you sure? (y/n)");  
 var choice = *readLine*()!!;  
 if(choice == "y") {  
 // delete video from all playlists  
 playlists.*forEach* **{ it**.deleteVideo(video) **}** hosting.deleteVideo(video);  
 }  
 }  
  
 override fun createPlaylist(){  
 *println*("Enter playlist name: ");  
 var name = *readLine*()!!;  
 playlists.add(playList(name));  
 }  
 override fun addVideoToPlayList(){  
 *println*("Enter playlist name: ");  
 var name = *readLine*()!!;  
 var playlist = playlists.*find* **{ it**.Name == name **}** if(playlist == null){  
 *println*("Playlist not found");  
 return;  
 }  
 *println*("Enter video name: ");  
 var videoName = *readLine*()!!;  
 var video = hosting.getVideo(videoName);  
 if(video == null){  
 *println*("Video not found");  
 return;  
 }  
 playlist.addVideo(video);  
 }  
  
  
 override fun getVideoList():List<Video>{  
 return hosting.getList();  
 }  
  
 override fun startPlaylist(Name:String){  
 var playlist = playlists.*find* **{ it**.Name == Name **}** if(playlist == null){  
 *println*("Playlist not found");  
 return;  
 }  
  
 playlist.startPlaylist();  
 }  
}

Файл VideoHosting.kt:

package hosting  
import videos.\*;  
  
abstract class VideoHostingBase {  
 abstract fun addVideo(video: Video)  
 abstract fun getVideo(Name:String): Video?  
 abstract fun getList():List<Video>  
 abstract fun deleteVideo(video: Video);  
}  
  
  
class VideoHosting: VideoHostingBase() {  
 private val videos = *mutableListOf*<Video>()  
  
  
  
 override fun addVideo(video: Video) {  
 videos.add(video)  
 }  
  
 override fun getVideo(Name:String): Video? {  
 val video = videos.*find* **{ it**.Name == Name **}** return video  
 }  
 override fun getList():List<Video> {  
 return videos  
 }  
  
 override fun deleteVideo(video: Video) {  
 videos.remove(video)  
 }  
  
}

Файл Videos.kt:

package videos  
abstract class playListBase(var Name:String){  
 abstract fun addVideo(video: Video)  
 abstract fun startVideo(video: Video)  
 abstract fun startPlaylist();  
 abstract fun deleteVideo(video: Video)  
}  
  
  
  
abstract class Video(var Name:String,var Duration:Int){  
 public fun Vatch(){  
 *println*("Watching video: $Name")  
 }  
}  
  
public class Cat\_Video(Name:String,Duration:Int):Video(Name,Duration){  
}  
  
class Dog\_Video(Name: String,Duration: Int) :Video(Name,Duration){  
}  
  
class Music\_Video(Name: String,Duration: Int) :Video(Name,Duration){  
}  
  
  
class playList(Name:String):playListBase(Name){  
 public val videos = *mutableListOf*<Video>()  
  
 override fun addVideo(video: Video) {  
 videos.add(video)  
 }  
  
 override fun startVideo(video: Video) {  
 *println*("Starting video: ${video.Name}")  
 }  
  
 private fun getNextVideo(video: Video): Video {  
 var index = videos.indexOf(video)  
 if(index == videos.size-1){  
 *println*("last video in playlist, starting from beginning")  
 return videos[0]  
 }  
 return videos[index+1]  
 }  
  
 override fun deleteVideo(video: Video) {  
 // chech if video is in playlist  
 videos.remove(video)  
 }  
 override fun startPlaylist(){  
 if(videos.size == 0){  
 *println*("Playlist is empty")  
 return  
 }  
 var video = videos[0]  
 startVideo(video)  
 while (true){  
 *println*("1. Next video")  
 *println*("2. Exit")  
 var choice = *readLine*()!!.*toInt*()  
 when(choice){  
 1->{  
 video = getNextVideo(video)  
 startVideo(video)  
 }  
 2->{  
 break  
 }  
 }  
 }  
 }  
}

Файл VideoHostingTest.kt:

package hosting  
  
import org.junit.jupiter.api.Test  
  
import org.junit.jupiter.api.Assertions.\*  
import videos.Cat\_Video  
  
class VideoHostingTest {  
  
 @Test  
 fun addVideo() {  
 var hosting = VideoHosting()  
 hosting.addVideo(Cat\_Video("1",10))  
 var video = hosting.getVideo("1")  
 assertEquals(video?.Name, "1")  
 }  
  
 @Test  
 fun getVideo() {  
 var hosting = VideoHosting()  
 hosting.addVideo(Cat\_Video("1",10))  
 var video = hosting.getVideo("1")  
 assertEquals(video?.Name, "1")  
 }  
  
 @Test  
 fun getList() {  
 var hosting = VideoHosting()  
 hosting.addVideo(Cat\_Video("1k5",10))  
 var videos = hosting.getList()  
 assertEquals(videos.size , 1);  
 }  
  
 @Test  
 fun deleteVideo() {  
 var hosting = VideoHosting()  
 var video = Cat\_Video("1",10);  
 hosting.addVideo(video)  
 var videos = hosting.getList()  
 hosting.deleteVideo(video)  
 assertEquals(videos.size , 0);  
 }  
  
}

Файл playListTest.kt:

package videos  
  
import org.junit.jupiter.api.Test  
  
import org.junit.jupiter.api.Assertions.\*  
  
class playListTest {  
  
  
 @Test  
 fun addVideo() {  
 var video = Cat\_Video("1",10)  
 var playlist = playList("1")  
 assertEquals(playlist.videos.size,0)  
 playlist.addVideo(video)  
 assertEquals(playlist.videos.size,1)  
 }  
 @Test  
 fun deleteVideo() {  
 var video = Cat\_Video("1",10)  
 var playlist = playList("1")  
 playlist.addVideo(video)  
 assertEquals(playlist.videos.size,1)  
 playlist.deleteVideo(video)  
 assertEquals(playlist.videos.size,0)  
 }  
 @Test  
 fun startPlaylist(){  
 var video = Cat\_Video("1",10)  
 var playlist = playList("1")  
 var res = playlist.startPlaylist()  
 assertEquals(res,Unit)  
 }  
}

Результат работы программы представлены на рисунках 1 – 8.

# 

Рисунок 1 – Результат выполнения программы

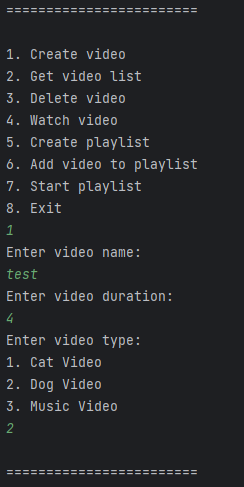


Рисунок 2 – Результат выполнения программы

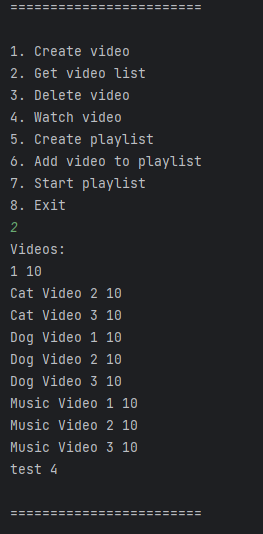


Рисунок 3 – Результат выполнения программы

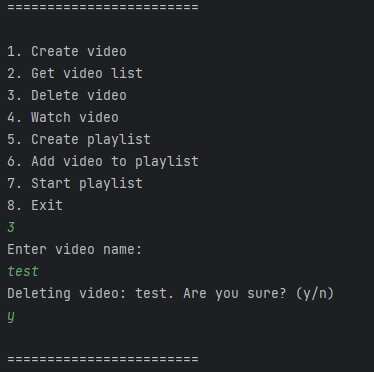


Рисунок 4 – Результат выполнения программы

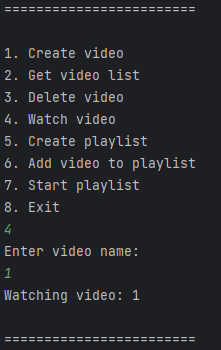


Рисунок 5 – Результат выполнения программы

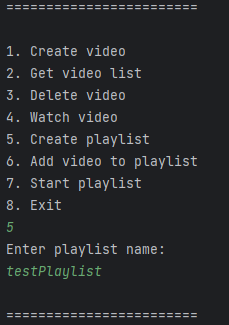


Рисунок 6 – Результат выполнения программы

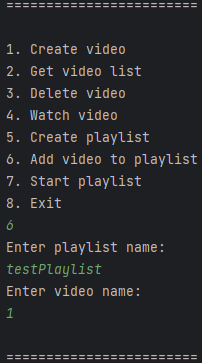


Рисунок 7 – Результат выполнения программы

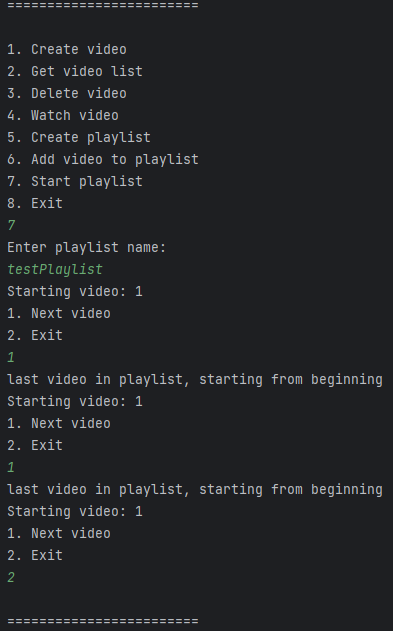


Рисунок 8 – Результат выполнения программы

Результат выполнения тестов представлен на рисунке 9.

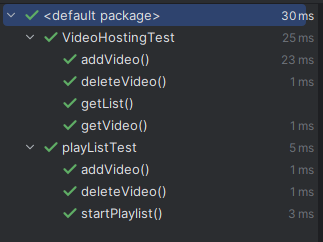


Рисунок 9 – Результат выполнения тестов

Вывод: в ходе выполнения лабораторной работы номер 1 по языку программирования kotlin мы познакомились с основами языка и научились на практике применять основные его конструкции. Было потрачено время на изучение UML диаграмм для корректного понимая задания лабораторной рабыты. При выполнении дополнительного задания познакомились с библиотекой Junit5 для написания тестов. При выполнении возникли трудности с понииманием работы конструкторов в Kotlin, в остальном работа выполнялась гладко. Затрачено часов – 6.