МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ СЕМЕНА КУЗНЕЦЯ

ЗАТВЕРДЖЕНО
на засіданні кафедри
інформаційних систем
Протокол № 1 від 22.08.2023 р.

ПОГОДЖЕНО
Проректор з навчально-методичної роботи

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ ІМЕНІ СЕМЕНА КУЗНЕЦЯ
№02071211

[illegible] Каріна НЕМАШКАЛО

МОБІЛЬНІ ТЕХНОЛОГІЇ

робоча програма навчальної дисципліни (РПНД)

Галузь знань	**12 "Інформаційні технології"**
Спеціальність	**121 "Інженерія програмного забезпечення"**
Освітній рівень	**перший (бакалаврський)**
Освітня програма	**"Інженерія програмного забезпечення"**

Статус дисципліни	**вибіркова**
Мова викладання, навчання та оцінювання	**англійська**

Розробник: к.т.н., доцент	підписано КЕП	Андрій ПОЛЯКОВ
Завідувач кафедри інформаційних систем	[illegible]	Дмитро БОНДАРЕНКО
Гарант програми	[illegible]	Олег ФРОЛОВ

Харків
2024

MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE
SIMON KUZNETS KHARKIV NATIONAL UNIVERSITY OF ECONOMICS

APPROVED
at the meeting of the department
information systems
Protocol No. 1 of 22.08.2023

AGREED
Vice-rector for educational and methodological work

[illegible] Karina NEMASHKALO

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ ІМЕНІ СЕМЕНА КУЗНЕЦЯ
02071211

MOBILE TECHNOLOGY

Program of the course

Field of knowledge	**12 "Information technologies"**
Specialty	**121 "Software engineering"**
Study cycle	**first (bachelor)**
Study programme	**"Software Engineering"**

Course status	**elective**
Language	**English**

Developer: Ph.D. (Technical sciences), associate professor	digital signature	Andrii POLIKOV
Head of Information systems department: Ph.D. (Technical sciences), associate professor	[illegible]	Dmytro BONDARENKO
Head of Study Programme: Ph.D. (Technical sciences), associate professor	[illegible]	Oleg FROLOV

Kharkiv
2024

INTRODUCTION

Mobile Technologies is a key course that covers the concepts and practices related to the development of mobile applications. Given the high level of penetration of mobile devices in society and their impact on our daily lives, the knowledge and skills related to mobile application development have become critical for information technology. The course is focused on learning the Kotlin programming language and the Android SDK as the main tools for creating applications on the Android platform. Students gain an understanding of the development process from initial design to testing and release of the application. They also study in detail the user interface, interaction with device sensors, networking, data storage, use of services, and much more. In addition, the course lays the foundations of application architecture to ensure reliable and productive software. Students gain practical experience in working with projects, which will help them to better absorb theoretical material.

The Mobile Technologies course involves mastering mobile application development skills, including: the ability to work with Android Studio and the Android SDK, understanding and using the Kotlin programming language, building an application interface using XML, using SQLite to store data in an application, integrating with web services via HTTP and REST, working with multimedia applications in Android, using Google Maps, and using the Git version control system. In addition, it is important to acquire the ability to develop applications that meet modern security, performance, and best design practices.

The purpose of teaching the course "Mobile Technologies" is to provide higher education students with a system of special theoretical knowledge and practical skills in the development of mobile applications on the Android platform. Development of analytical skills, teamwork skills, use of modern tools, development environments and methodologies for the development of business-oriented mobile applications. The study of educational material allows students to develop skills in the field of mobile technologies, to adapt to the rapidly changing conditions in the market of mobile applications and technologies.

Tasks of the course are:

– providing students with the necessary theoretical knowledge of the Android platform architecture, Kotlin programming language and various tools and libraries for developing Android applications;

– development of practical skills in developing mobile applications for the Android OS, including UI/UX design, Kotlin coding, use of databases and web services, work with multimedia and geolocation services;

– teaching students to use the Android SDK, the Android Studio development environment and the Git version control system for effective teamwork;

– awareness of the latest trends and best practices in mobile application development;

– development of skills in analysing and solving problems that arise during application development;

– preparing students for effective teamwork and independent project implementation;

– familiarising students with the process of deploying and testing mobile applications on various devices and platforms;

– developing a responsible attitude to the quality of code and application performance.

The subject matter of the course is a set of knowledge and skills required for the effective design, coding, testing and optimisation of mobile applications on the Android platform. This includes learning the Kotlin programming language, understanding the architecture of the Android platform, using tools and libraries to develop Android applications, and working with databases, services, the network, and the smartphone camera.

The object of the course is the processes of developing, testing, and making changes to mobile applications created to operate on the Android platform.

The learning outcomes and competencies formed by the course are defined in table 1.

Table 1

Learning outcomes and competences formed by the course

Learning outcomes	**Competences that must be mastered by a student of higher education**
LO07	GC05, SC13
LO12	SC03, SC14
LO17	GC02, SC12
LO18	GC06, SC12
LO21	SC08

where, LO07. To know and to apply in practice the fundamental concepts, paradigms, and basic principles of functioning of language, tool and computing software engineering tools.

LO12. Apply effective software design approaches in practice.

LO17. Be able to apply component-based software development methods.

LO18. Know and be able to apply information technologies for data processing, storage, and transmission.

LO21. To know, analyze, select, and competently apply information security (including cyber security) and data integrity tools in accordance with the applied tasks and software systems being developed.

GC02. Ability to apply knowledge in practical situations.

GC05. Ability to learn and master modern knowledge.

GC06. Ability to search, process and analyze information from various sources.

SC03. Ability to develop architectures, modules and components of software systems.

SC08. Ability to apply fundamental and inter disciplinary knowledge to successfully solve software engineering problems.

SC12. Ability to carry out the system integration process, apply change management standards and procedures to maintain the integrity, overall functionality and reliability of the software.

SC13. Ability to reasonably choose and master tools for software development and maintenance.

SC14. Ability to think algorithmically and logically.

COURSE CONTENT

Content module 1. Development environment for mobile devices. Kotlin language.

Topic 1. Introduction. Architecture and components of the Android mobile platform. Developer environment. GIT.

1.1. Introduction to the discipline. Control measures and learning outcomes.

1.2. Overview of Android and its architecture: introduction to Android: history, versions and developers of Android; Android architecture (Linux kernel, libraries, Android Runtime, Application Framework); main components of Android (Activities, Services, Broadcast Receivers, Content Providers); overview of Android Studio and its main functions; creating a simple application using Android Studio.

1.3. Introduction to Git and its basic commands: what is Git and why is it used? installing Git and setting up the environment; creating a new Git repository and cloning an existing one; creating commits, branches, and viewing the commit history; working with remote repositories (push, pull, fetch);

1.4. The concept of Android App Bundle and APK; what is an APK and how to create it; what is an Android App Bundle and how it differs from an APK; creating and verifying an Android App Bundle; using Split APK to distribute applications; optimising the size of APK and Android App Bundle.

1.5. Debugging an Android application: an introduction to debugging in Android Studio; setting up breakpoints; using the Debug window to track code execution; debugging when using connected devices; debugging memory issues and memory leaks using Android Studio.

1.6. Using Logcat to monitor application behaviour: introduction to Logcat and its tasks; using different levels of Log calls (Log.d, Log.i, Log.w, Log.e, Log.v, Log.wtf); filtering Logcat output using tags and importance levels; recording and exporting Logcat output; using Logcat to analyse exceptions and errors.

1.7. Android Gradle and its use: introduction to the Gradle build system; basic Gradle configuration files: build.gradle (project) and build.gradle (module); adding dependencies to an Android project; understanding Gradle scripts and the Gradle build lifecycle; using Gradle to create APKs and Android App Bundles.

Topic 2. Topic 2. Fundamentals of the Kotlin language. Basic constructions. Lambda. Classes. Idioms.

2.1. Introduction to Kotlin and its basic constructs: introduction to Kotlin: history, features and advantages; first Kotlin program; working with variables: var and val; basic data types and their operators; flow control: if-else, switch, loops.

2.2. Functions and lambda expressions in Kotlin: creating and calling functions; using function parameters and defining return types; using simple and block lambda expressions; lambda expressions with receivers and function types; collections and high-level functions: map, filter, reduce.

2.3. Classes and objects in Kotlin: creating and using a class and its objects; creating constructors and using properties; inheritance in Kotlin: superclasses and overrides; abstract classes, interfaces and their implementation, delegation; visibility restrictions (private, protected, internal, public); nested and inner classes.

2.4. Using null-safety in Kotlin: the basics of null-safety in Kotlin; working with variables that can be null; using the "??" safety operator, the "?:" operator, the "!!" operator and its consequences.

2.5. Kotlin idioms and good practices: an introduction to Kotlin idioms; working with null values (null safety); using the keywords "apply", "let", "it", "also" and "run"; using the "with" and "lazy" operators; destructive declarations and copying.

Topic 3. Kotlin language. Collections, filtering, transformations. Exceptions.

3.1. Working with collections in Kotlin: creating and using lists (List), sets (Set) and dictionaries (Map); operations on collections: adding, deleting, searching for items; immutable and mutable collections; creating and using mutable collections; walking through a collection using a for loop; using collections as function parameters.

3.2. Working with lists in Kotlin: creating lists, adding and removing items; processing lists: for loop, map, forEach methods; operations with lists (sorting, removing duplicates); converting lists (toSet(), toMap()); converting lists to strings and vice versa.

3.3. Filtering and transforming collections in Kotlin: filtering collections using the filter function; transforming collection elements using the map function; built-in methods for searching in collections (find, findLast, firstOrNull, lastOrNull); using a sequence of several operations through chaining; using flatMap and flatten to work with nested collections; using aggregate functions (reduce, fold, max, min, sum, average, etc.).

3.4. Working with exceptions in Kotlin: what are exceptions and why do you need to handle them; throwing exceptions; handling exceptions with try/catch blocks; finally block and its use; creating your own exceptions.

3.5. Creating and using extensions in Kotlin: extensions in Kotlin: what they are and how to use them; creating extension functions; creating extension properties; using extensions in real code; working with extensions of third-party classes.

Content module 2. Content module 2. Architecture and development tools for Android OS.

Topic 4. Screen markup: XML basics. Screen elements, element containers.

4.1. XML basics for Android screen markup: introduction to XML; using XML in Android for screen markup; XML syntax for describing screen elements; basic XML attributes for elements in Android; marking up the screen in Android Studio using XML.

4.2. Types of elements on the screen in Android: using text fields (TextView); working with buttons (Button); creating lists using RecyclerView; entering text using EditText; displaying images using ImageView.

4.3. Using containers to group elements on the screen: using LinearLayout to mark up the screen; working with RelativeLayout to create more complex interfaces; using FrameLayout to position elements on the screen; GridLayout and its features; ConstraintLayout to create flexible interfaces.

4.4. Adapt screen layout to different device sizes and orientations: respond to screen orientation changes; create different layouts for portrait and landscape orientations; use ConstraintLayout for responsive layout; use Android Studio Preview to view responsive layout; work with different dimensional resources.

4.5. Customising screen elements and styling the interface: changing the properties of elements: colours, fonts, and other attributes; creating and using styles and themes; visual styling of buttons, lists, and other elements; using vector and raster images; animating elements to the interface.

Topic 5. Accessing and managing screen elements from code. Handling events.

5.1. Accessing screen elements from code: using the findViewById method to access elements; using Android Kotlin Extensions to automatically generate display objects; working with screen element properties; handling exceptions when working with markup elements; using LiveData to monitor the state of UI components.

5.2. Interactive controls - Button, EditText, Checkbox and others: implementing a button and processing button clicks; using the text input field (EditText); using checkboxes and radio buttons (CheckBox, Switch); working with drop-down lists (Spinner); working with Radio Button and Radio Group.

5.3. User interaction events: processing click events; processing touch events; processing long click events; processing keyboard events; processing focus switching events.

5.4. Working with enumerators (RecyclerView): basics of working with RecyclerView; creating an Adapter for managing RecyclerView elements; handling click events on RecyclerView list items; optimising RecyclerView performance; implementing complex lists using RecyclerView.

5.5. Using animations to improve UX: the basics of the Android animation system; animation of changing the properties of elements (ObjectAnimator); spatial animations of transitions (Transition); animation of transitions between Activities (Activity Transition); using Animated Vector Drawables to create complex animations.

Topic 6. Application, Activity and its life cycle. Manifest. Android permission system.

6.1. General aspects of working with Applications and Activities in Android: the general structure of an Android application; the role and strategies for using the Application class; the concept and role of an Activity in an Android application; creating new Activities and integrating them into an application; methods for starting and ending Activities.

6.2. Activity lifecycle: understanding the Activity lifecycle; handling the initial stages of the lifecycle (onCreate, onStart); responding to pausing and resuming an Activity (onPause, onResume); handling the termination of the Activity lifecycle (onStop, onDestroy); using savedInstanceState to save the state of an Activity.

6.3. Android Manifest component in applications: understanding the role and use of the Android Manifest file; declaring an Activity in the manifest; using the manifest to control the screen orientation and other settings of the Activity; declaring optional access to hardware and services; using the manifest to declare permissions.

6.4. Working with permissions in Android: understanding the concept of permissions in Android; permission requirements outlined in the Android manifest; requesting permissions at runtime; handling scenarios where a user denies or accepts permissions; best practices for managing permissions.

6.5. Intents and working with them within an Activity: understanding the role of intents in Android; creating and using intents to launch an Activity; using intents to transfer data between Activities; responding to intents in an Activity; using intent filters to respond to system events.

Topic 7. Fragment and its life cycle. Android Jetpack Components: navigation.

7.1. General aspects of working with Fragment in Android: purpose and basic properties of Fragment; creating a new Fragment; integrating Fragment into Activity; interaction between Fragment and Activity; dynamically adding, removing, or replacing Fragment in Activity.

7.2. Fragment life cycle: a general overview of the Fragment life cycle; handling the creation of a Fragment: methods onAttach, onCreate, onCreateView; handling the transition of a Fragment to the activity state: methods onActivityCreate, onStart, onResume; handling the pause and stop of a Fragment: methods onPause, onStop; handling the destruction of a View and the Fragment itself: methods onDestroyView, onDestroy, onDetach.

7.3. Working with the Android Jetpack library and navigation components: general introduction to the Android Jetpack library and architectural components; introduction to the Navigation Component; organising navigation between Fragments using the Navigation Component; processing and transferring data between Fragments using the Navigation Component; responding to system events, such as the "Back" button, within the navigation.

7.4. Designing Responsive UIs with Fragments: designing a responsive UI using Fragments; arranging multiple Fragments in a single Activity; creating tabs or sliding screens using Fragments; creating dialogue boxes or full-screen advertising message boxes using DialogFragments; using Fragments to build menu navigation items.

7.5. Optimising data handling in Fragment: using ViewModel to manage data in Fragment; storing and restoring Fragment state; using architectural components to work with databases in Fragment; reusing View objects and display logic in different Fragments; using LiveData to monitor data in Fragment.

Topic 8. Data storage. Working with files. Preference. Basics of databases, Room.

8.1. Storing data in a file: introduction to the mechanism of storing data in a file; storing information in the internal memory of the device; storing information on an external memory card; reading and writing data to a file; security when storing data in a file.

8.2. Working with Shared Preferences: defining and working with Shared Preferences; writing data to Shared Preferences; reading data from Shared Preferences; editing and deleting data from Shared Preferences; saving complex data structures from Shared Preferences.

8.3. Introduction to SQLite databases: using SQLite in Android; creating a database and creating tables; inserting, updating, and deleting rows; querying a database: selecting and sorting data; closing a database and handling exceptions.

8.4. Using the Room ORM library: introducing and configuring the Room library; creating a database and entities using Room; creating a DAO to access the database; inserting, updating, and deleting records using Room; creating queries and processing the result.

8.5. Synchronisation with a remote database: working with remote APIs using Retrofit; using Gson to process JSON responses; working with OkHttp helper; using the Repository pattern for further asynchronous work with the database; using LiveData to monitor changes in the database.

Content module 3. Libraries and extension development tools for Android OS.

Topic 9. Access to the Internet. Retrofit, JSON.

9.1. Introduction to Internet access and HTTP requests: reviewing Android Internet usage scenarios; selecting network access rules; using Handler and Looper to solve network tasks; using HttpURLConnection for Internet requests; using AsyncTask for network requests.

9.2. General overview and use of Retrofit: introduction to Retrofit; configuring Retrofit to execute HTTP requests; configuring Enp-Point APIs and building requests with Retrofit; using Retrofit for synchronous and asynchronous calls; handling responses and errors with Retrofit.

9.3. Working with APIs and JSON: introduction to the JSON format; subscribe API End-Points that return JSON; deserialisation of JSON to Retrofit converters; serialisation of data to JSON for sending to the server; handling complex JSON responses.

9.4. Integrating OkHttp with Retrofit: using OkHttp as an HTTP client for Retrofit; logging HTTP requests and responses using OkHttp; configuring requests with additional headers or parameters; managing network contingency and request repetition; handling HTTP error responses using OkHttp.

9.5. The Gson Library for JSON Processing: Introduction to the Gson Library; converting JSON to Java objects using Gson; converting Java objects to JSON using Gson; using Gson annotations to accurately represent JSON; handling nested and arrayed objects in JSON using Gson.

Topic 10. Basics of application architecture. MVVM, ViewModel. LiveData, coroutine.

10.1. Introduction to application architecture: the importance of a well-structured application architecture; an overview of SOLID principles; an overview of the MVC pattern; an overview of the MVP pattern; an overview of the MVVM pattern.

10.2. Applying the MVVM pattern: an example of implementing the MVVM pattern in Kotlin; definition of ViewModel and its role in the MVVM architecture; definition of View and its role in the MVVM architecture; definition of Model and its role in the MVVM architecture; communication between components in MVVM.

10.3. Using ViewModel to save and manage UI data: introduction to the ViewModel class and its use; creating a ViewModel for an activity or fragment; saving and restoring data through a ViewModel; determining the life of a ViewModel in the context of the owner's life cycle; using ViewModel with LiveData.

10.4. Using LiveData for reactive programming: introduction to LiveData and its use in Android; creating and using LiveData objects; integrating LiveData with ViewModel; updating LiveData values and monitoring changes; using Transformations with LiveData.

10.5. Using coroutines for asynchronous tasks: an introduction to coroutines in Kotlin; comparing coroutines and callbacks/AsyncTask; creating and running simple coroutines; managing multithreading with coroutines: Dispatchers, withContext; error handling and exceptions in the context of coroutines.

Topic 11. Services, WorkManager. Notifications.

11.1. Understanding and using services: an introduction to Android services; creating a simple service; creating a Bound Service; starting and stopping a service; synchronising data in the background using services.

11.2. Working with WorkManager: introduction to WorkManager; creating a simple task (WorkRequest); running a task (WorkRequest); saving the status and processing the results of tasks; managing an unstable connection and using periodic requests.

11.3. Introduction to notifications: understanding and benefits of notifications; creating and configuring a notification channel; creating and broadcasting a notification; opening an activity through a notification; configuring actions and buttons in a notification.

11.4. Creating persistent notifications: general ideas and uses of persistent notifications; creating a notification that cannot be deleted or postponed; setting audio and intensity settings for notifications; adding a large image or text to a notification; sending a notification that requires an immediate user response.

11.5. Working with Foreground services: introduction to foreground services; creating a foreground service; switching a service to foreground mode; updating a foreground service notification; stopping or terminating a foreground service.

Topic 12. Working with GPS, working with maps, working with the camera.

12.1. Using GPS and getting geolocation: introduction to GPS and geolocation services in Android; requesting geolocation permissions; getting the current location of the user; working with the Location object; location change listers.

12.2. Working with maps using the Google Maps API: an introduction to the Google Maps API; adding a Google Map to your application; setting up and using a GoogleMap object; adding markers and drawings to a map; camera movement and controlling the view.

12.3. Working with the camera: an introduction to using the camera in Android; requesting camera permissions; launching the camera's Intetn to take a photo; taking and saving a photo; using the Android Camera API for more control.

12.4. Connecting to and using Google Play Services: general knowledge of Google Play Services; installing and connecting Google Play Services to the project;

identifying and resolving connectivity issues; using Google Play Services geolocation services; using other Google Play Services APIs such as Google Drive API, Google Fit API, etc.

12.5. Advanced camera control using the Camera2 API: an introduction to the Camera2 API; working with the CameraDevice object; creating a preview request; setting up the CaptureRequest object for taking photos; working with turning on/off the flash, focusing, and other camera settings.

The list of laboratory studies in the course is given in table 2.

Table 2

The list of laboratory studies

Name of the topic and/or task	Content
Topic 1 – 2	Study of development tools in Kotlin. Basic constructs of the Kotlin language. GIT code version control
Topic 3	Study of OOP and lambda in Kotlin
Topic 4 – 6	Deploying the Android Studio development environment and studying the basic elements of Activity, lifecycle, and inter-component communication
Topic 5 – 7	Exploring the use of RecyclerView lists in a mobile application and the use of snippets and navigation between them
Topic 8	Investigating the operation of databases on Android using Room Persistence Library
Topic 9	Exploring the operation of web services with Retrofit and processing JSON responses
Topic 10 – 11	Creating asynchronous services using Kotlin Coroutines and LiveData with ViewModel. Storing and transferring Activity state when the device is rotated.
Topic 12	Exploring how Goole Map, GPS, and camera work

The list of self-studies in the course is given in table 3.

Table 3

List of self-studies

Name of the topic and / or task	Content
Topic 1 – 12	Study of lecture material
Theme 1 – 12	Preparation for laboratory classes
Theme 1 – 12	Preparation for current tests
Theme 1 – 12	Preparation for the exam

The number of hours of lectures and laboratory studies and hours of self-study is given in the technological card of the course.

TEACHING METHODS

In the process of teaching the course, in order to acquire certain learning outcomes, to activate the educational process, it is envisaged to use such teaching methods as:

Verbal (lecture (Topics 1 - 3, 4, 7, 9, 12), problem lecture (Topics 1, 4, 6 - 8), lecture-visualisation (Topics 1 - 12)).

Visual (demonstration (Topics 1 - 12)).

Laboratory work (Topics 1 - 12), case method (Topics 4, 6 - 8, 9 - 12).

FORMS AND METHODS OF ASSESSMENT

The University uses a 100-point cumulative system for assessing the learning outcomes of students.

Current control is carried out during lectures, laboratory classes and aims to check the level of readiness of the higher education applicant to perform specific work and is assessed by the amount of points scored:

– for courses with a form of semester control as an exam: maximum amount is 60 points; minimum amount required is 35 points.

Final control includes current control and an exam.

Semester control is conducted in the form of a semester exam.

The maximum amount of points that a higher education student can receive during an exam is 40 points. The minimum score for an exam to be considered passed is 25 points.

Final grade in the discipline is determined by summing up the points for the current and final control.

During the teaching of the course, the following control measures are used:

Current control: defence of laboratory works (40 points), current control works (10 points), presentations (10 points).

Semester control: Exam (40 points)

More detailed information on the assessment system is provided in technological card of the course.

An example of an exam card and assessment criteria.

An example of an exam card

Semyon Kuznets Kharkiv National University of Economics
First (bachelor) level of higher education
Specialty "Software Engineering"
Educational program "Software engineering"
Semester I
Course "Mobile Technology"

EXAM CARD No. 1

Task 1. Test.

Question No. 1	
Which statement in the build.gradle file correctly indicates that the corresponding module is an Android library module?	
Choose one of 4 answers:	
1)	apply plugin: 'com.module.library'
2)	apply plugin: 'com.android.library'
3)	apply plugin: 'com.module.library'
4)	include plugin: 'com.module.library'

Question No. 2	
What is missing from the activity life cycle?	
Choose one of 4 answers:	
1)	onPause()
2)	onResume()
3)	onOpen()
4)	onStart()

Question No. 3	
True or false? When your app goes into the background, it is not guaranteed to be destroyed. It can only stop and wait for the user to return.	
Choose one of 2 answers:	
1)	True
2)	False

Question No. 4	
True or false? When the enabled attribute is used, it determines whether the view is visible.	
Choose one of 2 answers:	
1)	False
2)	True

Question No. 5	
What is the correct syntax to print "Hello World" in Kotlin?	
Choose one of 4 answers:	
1)	println("Hello World")
2)	Console.WriteLine("Hello World")
3)	cout << "Hello World";
4)	System.out printline("Hello World");

Question No. 6	
How to start writing a **while loop in Kotlin?**	
Choose one of 4 answers:	
1)	while (x < y)
2)	if x > y while
3)	while x < y
4)	while x < y then

Question No. 7	
What is to in the example below: **val test = 33 to 42**	
Choose one of 4 answers:	
1)	Kotlin keyword to create Pair(33, 42)
2)	Syntax error
3)	Kotlin keyword to create a Range from 33 to 42
4)	An infix expansion method that creates a Pair(33, 42)

Question No. 8	
Which function declaration is correct in Kotlin?	
Choose one of 4 answers:	
1)	int sum(int a, int b)
2)	function sum(a: Int, b: Int): Int
3)	int sum(a: Int, b: Int)
4)	fun sum(a: Int, b: Int): Int

Question No. 9	
Specify the option used to handle null exceptions in Kotlin?	
Choose one of 4 answers:	
1)	Range
2)	Sealed Class
3)	Elvis Operator
4)	Lambda function

Question No. 10	
What is the default behaviour of Kotlin classes?	
Choose one of 4 answers:	
1)	All classes are public
2)	All classes sealed
3)	All classes final
4)	All classes abstract

Task 2.

Develop a mobile application that consists of a single screen form and displays information about one object of the corresponding subject area. The application should load and save data to a file.

The application must provide (use):

constrained layout, which allows you to create complex user interfaces by using constraints between different elements;

restrict the virtual keyboard in accordance with the data entered;

store and restore data when switching to the background and using data saving to the internal file storage for permanent storage in the JSON serialisation format;

call a short Toast message about saving data;

use the button with the usual hold (Click) to open a web page in the browser;

internationalisation: English and Ukrainian languages;

use the GIT code versioning system. All implementations of the requirements should be documented in the form of commits. Upload (push) to the gitlab.hneu.net server to the **examen** project created in your account.

Approved at the meeting of the Department of
Information systems protocol No. __ from «___» __________20___р.

Examiner	__________________	Ph.D., associate professor, Andrii POLIKOV
Head of the Department	__________________	Ph.D., associate professor. Dmytro BONDARENKO

Evaluation criteria

The final score for the exam is the sum of the scores for all tasks, rounded to the nearest whole number according to the rules of mathematics.

The algorithm for solving each task includes separate stages that differ in complexity, labour intensity and importance for solving the task. Therefore, individual tasks and stages of their solution are evaluated separately from each other as follows:

Task 1 (test).

The first question is dedicated to testing theoretical knowledge of the discipline. The test includes 10 questions worth 1.5 points each. A total of 15 points. The test lasts 20 minutes.

Task 2 (heuristic).

The second question is about the development of a mobile application in relation to the task and provide a software project with the program code and a report with screenshots demonstrating the application. The project is developed in the Kotlin programming language and the Android platform. The applicant must create a project in the Android Studio environment. The main goal of this task is to test the applicant's practical skills in designing the application architecture, designing the user interface, and being able to implement business logic in the application. In doing so, the applicant is allowed to use existing reference literature. After checking the application, the applicant receives K_1 points for the following requirements (Table 4).

Table 4

Assessment criteria for the diagnostic task

The second task is scored from 0 to 25 points. The number of points for the second task is determined in accordance with the following scale:

Points	Criterion.
25	The task has been completed in full.
24	The task is completed in full. There are small comments on the organisation of the code structure and interface.

Points	Criterion.
22-23	The task is mostly completed. There are comments on the organisation of the code structure and interface.
20-21	The task is completed, but not in full. The program works, but some requirements or features specified in the assignment are not implemented.
18-19	The task is completed, but not in full. The programme works, but two or three requirements or features specified in the assignment are not implemented.
16-17	The task is completed, but not in full. The programme is working, but four or five requirements or capabilities specified in the assignment have not been implemented.
11-15	The task is completed, but not in full. The programme works, but more than five requirements or features specified in the assignment are not implemented.
7-10	The programme at least allows you to perform one action (capability) correctly.
3-6	The programme runs, but contains gross errors, no task or requirement is fully completed.
2	The programme does not correspond to the task statement.
1	The programme does not contain the programme code developed by the student.
0	The programme has obvious signs of non-independence of its development.

RECOMMENDED LITERATURE

Basic

1. Griffiths D. Head first Kotlin / D. Griffiths, D. Griffiths. — Beijing Boston Farnham Sebastopol Tokyo : O’Reilly, 2019. — 448 p.
2. Griffiths D. Head First Android Development: A Learner's Guide to Building Android Apps with Kotlin / D. Griffiths, D. Griffiths. — Beijing Boston Farnham Sebastopol Tokyo : O’Reilly, 2021. — 890 p.
3. Sills B. Android programming: The Big Nerd Ranch guide / B. Sills, B. Gardner, K. Marsicano, C. Stewart. — Atlanta, GA : Big Nerd Ranch, 2022. — 667 p.
4. Поляков А. О Аналіз методів і технологій розроблення мобільних додатків для платформи Android : навчальний посібник [Електронний ресурс] / Поляков А. О, Федорченко В. М, Шматко О. В. — Харків : ХНЕУ імені С. Кузнеця, 2017. — 286 р. URL: http://repository.hneu.edu.ua/handle/123456789/20105

Additional

5. Jemerov D. and others Kotlin in Action / D. Jemerov, S. Isakova. — Shelter Island, NY : Manning Publications Co, 2017. — 334 p.
6. Laurence P.-O. Programming Android with Kotlin: Achieving Structured Concurrency with Coroutines / P.-O. Laurence, A. Hinchman-Dominguez, G. B. Meike, M. Dunn. — Beijing Sebastopol, CA : O’Reilly, 2022. — 336 p.

7. Tigcal J. Simplifying Android Development with Coroutines and Flows / J. Tigcal. — Birmingham, UK : Packt Publishing, Limited, 2022. — 159 p.

8. Muschko B. Gradle in action / B. Muschko. — Shelter Island, NY : Manning, 2014. — 456 p.

9. Tuominen T. RxJava for Android Developers / T. Tuominen. — Shelter Island, NY : Manning Publications Co., 2019. — 514 p.

10. Федорченко В. Аналіз ефективності технологій розроблення мобільних застосунків для ОС Android / В. Федорченко, А. Поляков, О. Сєвєрінов // Вісник ХНАДУ — 2022. — Вип. 96. — P. 81–90.

Information resources

11. Kotlin Docs | Kotlin [Електроний ресурс] // Kotlin Help. — Електрон. дані. — Режим доступу: https://kotlinlang.org/docs/home.html.

12. Download Android Studio & App Tools - Android Developers [Електроний ресурс] // Електрон. дані. — Режим доступу: https://developer.android.com/studio.

13. Android Compose Tutorial | Jetpack Compose | Android Developers [Електроний ресурс] // Android Developers. — Електрон. дані. — Режим доступу: https://developer.android.com/jetpack/compose/tutorial.

14. Design for Safety | App quality | Android Developers [Електроний ресурс] // Android Developers. — Електрон. дані. — Режим доступу: https://developer.android.com/quality/privacy-and-security.

15. Fragments | Android Developers [Електроний ресурс] // Android Developers. —Електрон. дані. — Режим доступу: https://developer.android.com/guide/fragments.

16. Fundamentals of testing Android apps | Android Developers [Електроний ресурс] // Android Developers. —Електрон. дані. — Режим доступу: https://developer.android.com/training/testing/fundamentals.

17. Services overview | Background work [Електроний ресурс] // Android Developers. — Електрон. дані. — Режим доступу: https://developer.android.com/develop/background-work/services.

18. Мобільні технології (6.04.121) [Електронний ресурс] / Розробники Андрій Поляков, Володимир Федорченко // Персональні навчальні системи ХНЕУ ім. С. Кузнеця — Електрон. дані. — Режим доступу: https://pns.hneu.edu.ua/course/view.php?id=1366.