МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «ВГУ»)

УТВЕРЖДАЮ

Заведующий кафедрой
Кибербезопасности информационных систем
Кенин С. Л.
22.03.2024 г.

РАБОЧАЯ ПРОГРАММА УЧЕБНОЙ ДИСЦИПЛИНЫ
Б1.В.03 Создание мобильных приложений iOS

1. Код и наименование направления подготовки:

02.04.02 Фундаментальная информатика и информационные технологии

2. Профиль подготовки: Технологии разработки мобильных приложений

3. Квалификация выпускника: магистр

4. Форма обучения: очная

5. Кафедра, отвечающая за реализацию дисциплины: МО ЭВМ

6. Составители программы: Болотова Светлана Юрьевна,
кандидат физико-математических наук, доцент
7. Рекомендована: НМС факультета ПММ, протокол № 5 от 22.03.2024

8. Учебный год:_ 2025/2026 Семестр: 3

9.Цели и задачи учебной дисциплины
Целями освоения дисциплины являются: углубленное изучение принципов и получение практических навыков программной инженерии в области разработки программного обеспечения для мобильных устройств.
Основные задачи преподавания дисциплины следующие:
углубленное изучение мобильной операционной системы iOS;
получение практических навыков по разработке полноценного мобильного приложения с применением всех изученных принципов, методик, методов и средств разработки мобильных приложений.

10. Место учебной дисциплины в структуре ООП: Дисциплина относится к части, формируемой участниками образовательных отношений, Блока 1. Дисциплины (модули). Изучение курса должно базироваться на знании обучающимися материала курса «Программирование на платформе iOS». Дисциплина является базовой для изучения курсов «Безопасность мобильных устройств».

11. Планируемые результаты обучения по дисциплине/модулю (знания, умения, навыки), соотнесенные с планируемыми результатами освоения образовательной программы (компетенциями) и индикаторами их достижения:

Код	Название компетенции	Код(ы)	Индикатор(ы)	Планируемые результаты обучения
ПК-4	Способен разрабатывать профессионально-ориентированные программные средства и приложения на основе мобильных технологий	ПК-4.1	Использует методы и приемы алгоритмизации поставленных задач с учетом возможностей современных мобильных технологий	Знать: методы и приемы алгоритмизации поставленных задач с учетом возможностей современных мобильных технологий Уметь: использовать использует методы и приемы алгоритмизации поставленных задач с учетом возможностей современных мобильных технологий Владеть: навыками использования методов и приемов алгоритмизации поставленных задач с учетом возможностей современных мобильных технологий
ПК-5	Способен выбирать технологии и средства разработки мобильных приложений, определять ключевые сценарии для архитектуры мобильных приложений, разрабатывать новые алгоритмические, методические и технологические решения в сфере разработки мобильных приложений	ПК-5.1 ПК-5.2 ПК-5.3	Владеет основами проектирования, знает элементы архитектурных решений информационных систем, технологии и средства разработки программного обеспечения. Проектирует архитектуру, оценивание ПО, применяет в практической деятельности профессиональные стандарты в области информационных технологий. Имеет практический опыт в выборе технологий и средств разработки ПО, определяет цели, предположения и ограничения.	Знать: элементы архитектурных решений информационных систем, технологии и средства разработки программного обеспечения Уметь: проектировать архитектуру, оценивать ПО, применять в практической деятельности профессиональные стандарты в области информационных технологий. Владеть: выбором технологий и средств разработки ПО, определением целей, предпочтений и граничений.

12. Объем дисциплины в зачетных единицах/час. — 5/180.

Форма промежуточной аттестации: экзамен

13. Трудоемкость по видам учебной работы

Вид учебной работы	Трудоемкость	
		По семестрам

		Всего	3 семестр		
Контактная работа					
в том числе:	лекции		16		
	практические				
	лабораторные		32		
Самостоятельная работа			96		
Промежуточная аттестация		Экзамен	36		
Итого:			180		

13.1. Содержание дисциплины

п/п	Наименование раздела дисциплины	Содержание раздела дисциплины	Реализация раздела дисциплины с помощью онлайн-курса, ЭУМК
1. Лекции			
1.1	Многопоточность	Рассматривается использовании многопоточности только в одной важной нише, связанной с НЕ блокировкой UI.	https://edu.vsu.ru/course/view.php?id=18174
1.2	Жесты. JSON	Рассматривается пара тем (UserDefaults API и JSON encoding/decoding), связанных с постоянным хранением (persistence), а также - концептуальные основы обработки «жестов» мультитач (multi-touch gesture).	https://edu.vsu.ru/course/view.php?id=18174
1.3	Data Flow.	Обсуждаются «Обертки Свойств», Property Wrappers, (типа @Published, @EnvironmentObject, @Binding) обсуждаются наряду с «издателями» Publishers. Затем EmojiArt использует их для автосохранения и для поддержки выбора между множеством палитр с эмоджи .	https://edu.vsu.ru/course/view.php?id=18174
1.4	Modal Presentation и Navigation	Расширение возможностей UI с помощью методов модального представления и навигации. Получение текста от пользователя через TextField. Понимание ТИПа KeyPath. Хранение множества EmojiArt документов.	https://edu.vsu.ru/course/view.php?id=18174
1.5	Core Data	UserDefaults. Простая. Ограниченная (только Property Lists). Очень небольшая по объему. Codable / JSON. Понятный способ преобразования практически любой структуры данных в формат JSON обмена данными и наоборот. UIDocument. Core Data Мощная. Объектно-ориентированная. Элегантная интеграция со SwiftUI. Cloud Kit. Запоминание данных в базе данных, расположенной в «облаке» (то есть в интернете), которая работает полностью асинхронно. Следовательно, данные появляются на любых устройствах пользователя. Это очень простая в использовании база данных, у нее есть основные операции базы данных, но не такие полноценные как у Core Data, однако она отлично работает с Core Data (так что ваши данные в Core Data могут также появиться на всех устройствах) FileManager/URL/Data. Запоминание данных	https://edu.vsu.ru/course/view.php?id=18174

		в Unix файловой системе в iOS.	
1.6	Интеграция UIKit	Интеграция пре-SwiftUI iOS кода в SwiftUI.	https://edu.vsu.ru/course/view.php?id=18174
2. Лабораторные работы			
2.1	Многопоточность	Идет разработка совершенно нового демонстрационного примера EmojiArt, начиная с обзора MVVM, а затем с использованием таких API, как ScrollView, UIImage и Drag & Drop.	https://edu.vsu.ru/course/view.php?id=18174
2.2	Постоянное хранение и жесты	Усовершенствуется демонстрационный пример EmojiArt в плане постоянного хранения (persistence) внесенных изменений и поддержки мультитач «жестов» pinching (жест сведения и разведения пальцев), используемого для масштабирования, и pan (движение пальца по экрану) для перемещения документа по экрану.	https://edu.vsu.ru/course/view.php?id=18174
2.3	Enroute Picker	Разработать UI, с помощью которого можно будет осуществлять фильтрацию рейсов по аэропорту прибытия destination, по аэропорту отправления origin, по авиакомпании airline и по тому, находятся рейсы уже в воздухе или еще ожидают вылета на земле inTheAir.	https://edu.vsu.ru/course/view.php?id=18174
2.4	Core Data	Пример добавляет базу данных Core Data для Enroute.	https://edu.vsu.ru/course/view.php?id=18174

13.2. Темы (разделы) дисциплины и виды занятий

№ п/п	Наименование темы (раздела) дисциплины	Виды занятий (часов)				
		Лекции	Практические	Лабораторные	Самостоятельная работа	Всего
1	Многопоточность	2		6	12	20
2	Жесты. JSON	2			12	14
3	Data Flow.	2			12	14
4	Modal Presentation и Navigation	4			12	16
5	Core Data	4		8	12	24
6	Интеграция UIKit	2		6	12	20
7	Постоянное хранение и жесты			6	12	18
8	Enroute Picker			6	12	18
9	Подготовка к экзамену	0	0	0	36	36
	Итого:	16		32	132	180

14. Методические указания для обучающихся по освоению дисциплины

Указание наиболее сложных разделов, работа с конспектами лекций, презентационным материалом. При использовании дистанционных образовательных технологий и электронного обучения выполнять все указания преподавателей по работе на LMS-платформе, своевременно подключаться к online-занятиям, соблюдать рекомендации по организации самостоятельной работы.

15. Перечень основной и дополнительной литературы, ресурсов интернет, необходимых для освоения дисциплины (список литературы оформляется в соответствии с требованиями ГОСТ и используется общая сквозная нумерация для всех видов источников)

а) основная литература:

№ п/п	Источник
1	Ваттер, А. П. IPAD, IPHONE, MACBOOK и сервисы APPLE. Все о совместном использовании : руководство / А. П. Ваттер. — Санкт-Петербург : Наука и Техника, 2016. — 256 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/74666.

2	Вейн, Ч. Swift подробно / Ч. Вейн ; перевод с английского Д. А. Беликова. — Москва : ДМК Пресс, 2020. — 422 с. — ISBN 978-5-97060-780-0. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/131722.

Б) дополнительная литература:

№ п/п	Источник
1	Разработка и продажа программ для iPhone и iPad / Дмитрий Елисеев. – БХВ- Петербург, 2012. ISBN 978-5-9775-0687-8.
2	SWIFT: БАЗОВОЕ РУКОВОДСТВО [Электронный ресурс] / М. Цукалос // Linux Format (Линукс Формат) .— 2016 .— №4 .— С. 82-85 .— Режим доступа: https://rucont.ru/efd/566911

В) информационные электронно-образовательные ресурсы (официальные ресурсы интернет)*:

№ п/п	Ресурс
1	https://edu.vsu.ru/course/view.php?id=18174
2	https://cs193p.sites.stanford.edu Курс лекций Ipad and Iphone Application Development

* Вначале указываются ЭБС, с которыми имеются договора у ВГУ, затем открытые электронно-образовательные ресурсы, онлайн-курсы, ЭУМК

16. Перечень учебно-методического обеспечения для самостоятельной работы (учебно-методические рекомендации, пособия, задачники, методические указания по выполнению практических (контрольных), курсовых работ и др.)

№ п/п	Источник
1	Соколова В. В. Разработка мобильных приложений : учебное пособие. — Томск: Изд-во ТПУ, 2014. — 175 с.: ил.
2	Нахавандипур В. iOS. Приемы программирования. — Санкт- Петербург: Питер, 2014. — 832 с.

17. Образовательные технологии, используемые при реализации учебной дисциплины, включая дистанционные образовательные технологии (ДОТ), электронное обучение (ЭО), смешанное обучение):

Для реализации учебного процесса используется бесплатная полнофункциональная интегрированная среда разработки XCode. Дисциплина реализуется с применением электронного обучения и дистанционных образовательных технологий. Для организации занятий рекомендован онлайн-курс «Программирование для платформы iOS», размещенный на платформе Электронного университета ВГУ (LMS moodle), а также Интернет-ресурсы, приведенные в п.15в.

18. Материально-техническое обеспечение дисциплины:

Учебная аудитория для проведения лекций, практических занятий, организации самостоятельной работы,
проведения текущих и промежуточных аттестаций: специализированная мебель, доска маркерная или
меловая, компьютер (ноутбук), мультимедийное оборудование (проектор, экран, средства звуковоспроизведения), допускается использование переносного оборудования.
Для самостоятельной работы необходимы компьютерные классы, помещения, оснащенные компьютерами с доступом к сети Интернет.
Программное обеспечение: Xcode
Материально-техническое обеспечение:
Моноблок Apple iMac MD093RU/A (14 шт.): процессор Intel Core i5 (2.70 GHz), оперативная память 8 Гб, HDD 1 Тб, видеокарта GeForce GT640M 512Мб, диагональ экрана 21,5"
Компьютер APPLE Mac Pro MD772RU/A Xeon W3565 в составе:
системный блок APPLE: процессор Intel Xeon W3565, оперативная память 8Гб, HDD 2Тб, видеокарта AMD Radeon HD 5770
Коммутатор HP ProCurve Switch 1400-24G

Мультимедиа-проектор BENQ MH535
Доска магнитно-маркерная на стенде (100х150см), 2-сторонняя, BRAUBERG PREMIUM

19. Оценочные средства для проведения текущей и промежуточной аттестаций

Порядок оценки освоения обучающимися учебного материала определяется содержанием следующих разделов дисциплины:

№ п/п	Наименование раздела дисциплины (модуля)	Компетенция(и)	Индикатор(ы) достижения компетенции	Оценочные средства
1.	Многопоточность	ПКВ-5	ПК-4.1	Лабораторная работа
2.	Жесты. JSON	ПКВ-5	ПК-5.1	Лабораторная работа
3.	Data Flow.	ПКВ-5	ПК-5.2	Лабораторная работа
4.	Modal Presentation и Navigation	ПКВ-5	ПК-4.1 ПК-5.3	Лабораторная работа
5.	Core Data	ПКВ-5	ПК-5.2 ПК-5.3	Лабораторная работа
6.	Интеграция UIKit	ПКВ-5	ПК-5.2 ПК-5.3	Лабораторная работа
7.	Постоянное хранение и жесты	ПКВ-5	ПК-4.1 ПК-5.1 ПК-5.2 ПК-5.3	Лабораторная работа
8.	Enroute Picker	ПКВ-5	ПК-5.2 ПК-5.3	Лабораторная работа
Промежуточная аттестация форма контроля - экзамен				Экзаменационные билеты

20 Типовые оценочные средства и методические материалы, определяющие процедуры оценивания

20.1 Текущий контроль успеваемости

Контроль успеваемости по дисциплине осуществляется с помощью следующих оценочных средств: лабораторная работа

Примеры лабораторных работ

Задание 1. Загрузите версию EmojiArt с Лекции. Не ломайте ничего, что там работает, как часть вашего решения этого задания.
Поддержите выбор одного или нескольких эмоджи, которые были “перетянуты” на ваш EmojiArt документ (то есть вы выбираете эмоджи в документе, а не на палитре в верхней части экрана). Вы можете показать, какие эмоджи были выбраны, любым способом. Этот выбор не требует постоянного хранения (другими словами, повторный запуск вашего приложения не сохранит этот выбор.
Выполнение жеста Tap на невыбранном эмоджи должно выбирать этот эмоджи.
Выполнение жеста Tap на выбранном эмоджи должно делать этот эмоджи не выбранным.
Однократный жест Tap на фоновом изображении вашего EmojiArt (то есть один раз кликнуть за исключением эмоджи) должен сделать не выбранными все эмоджи.
“Перетаскивание” выбранных эмоджи должно перемещать всю выборку эмоджи вслед за пальцами пользователя.
Если пользователь выполняет жест Drag, когда ничего не выбрано, то перемещайте (pan) полный документ.
Если пользователь выполняет жест Pinch где-то на EmojiArt документе и есть выбор эмоджи, то все выбранные эмоджи должны масштабироваться в соответствии с жестом Pinch.
Если нет выбора эмоджи во время выполнения жеста Pinch, то выполняется масштабирование полного документа.
Сделайте возможным удаление эмоджи из EmojiArt документа. В этом обязательном пункте Задания намеренно не указывается, какое UI действие должно вызывать удаление эмоджи. Будьте креативны и попытайтесь найти способ удаления эмоджи, который был бы комфортабельным и интуитивным.

Задание 2. Уберите опцию “случайного выбора числа карт” для темы (theme) из вашей игры Memorize. Теперь каждая тема (theme) будет иметь своё собственное, заранее определенное число карт. Другими словами. Сколько карт в игре - это часть темы (theme) для этой игры и не может быть больше “случайным”. Каждый раз, когда стартует новая игра, печатайте print JSON представление темы (theme), используемой в данный момент на консоли. Все элементы темы (theme) (её имя, набор эмоджи для выбора, сколько пар карт в игре и цвет темы) должны быть включены.

Задание 3. Теперь ваше приложение Memorize должно при запуске отображать UI «выбора темы». Примеры см. в прикрепленных изображениях, но вы можете проявить творческий подход к тому, как показывать каждую тему.
Используйте List для показа тем (themes).
Каждая строка в List показывает имя темы (theme), цвет темы, сколько карт в теме и некоторую выборку эмоджи.
Касание темы (theme) в List переводит (так как List находится внутри NavigationView) вас на игру с этой темой.
Во время игры название темы должно где-то отображаться на экране, и вы также должны продолжать поддерживать существующие функции, такие как счет, новая игра и т. д. (Но вы можете изменить UI, если хотите).
Это нормально, если при переходе от игры обратно к выбору игры в List, а затем обратно к текущей игре в полном разгаре все-таки игра перезапускается, хотя разумные реализации, вероятно, этого не сделают (кроме случаев, когда тема (theme) изменена (см. Ниже), поскольку это в этом случае вы все равно захотите перезапустить игру.
Предоставьте некоторый UI для добавления новой темы (theme) в список тем List.
View для выбора темы (theme) должен поддерживать режим редактирования Edit Mode, при котором вы можете уничтожать темы (themes) и при котором у вас есть доступ к некоторому UI (то есть кнопке Button или изображении Image в каждой строке), который выводит вас модально (через sheet или popover) на UI Редактора Темы и дает возможность редактировать эту тему (theme).
Редактора Темы должен быть формой Form.
В Редакторе Темы пользователю разрешается редактировать имя темы (theme), добавлять эмоджи к теме, удалять эмоджи из темы и сколько карт в этой теме. (Есть Дополнительный Пункт, который связан с возможностью редактирования цвета темы).
Темы (themes) должны постоянно сохраняться (перезапуск вашего приложения не должно вызывать потерю всех отредактированных тем).
Ваш UI должен хорошо выглядеть как на iPhone, так и на iPad.
Заставьте ваше приложение работать на любом физическом устройстве iOS.

Описание технологии проведения

Каждая лабораторная работа выполняется на основе задания и соответствующей лекции. После выполнения задания на лабораторную работу каждый студент должен выполнить те же действия, но уже по своей теме, которая относится к домашнему заданию по дисциплине. Таким образом, после каждой лабораторной работы формируются необходимые части/знания для выполнения домашнего задания.

Требования к выполнению заданий (или шкалы и критерии оценивания)
Каждая лабораторная работа оценивается по принципу «зачет/незачет»

«Зачет» ставится, если сделано верно не менее 80% задания
«Незачет» ставится, если сделано верно менее 80% задания

20.2 Промежуточная аттестация

Промежуточная аттестация по дисциплине осуществляется с помощью следующих оценочных средств:
Собеседование по экзаменационным билетам (по билетам к зачету)

Пример экзаменационного билета

1. Обертки Свойств
2. Интеграция в Swift UI

Описание технологии проведения

Экзамен проходит в письменной форме

Требования к выполнению заданий, шкалы и критерии оценивания

Критерии оценки:

оценка «отлично» выставляется обучающемуся, если объем правильного ответ не менее чем 75%;

оценка «хорошо» выставляется обучающемуся, если объем правильного ответ не менее чем 50%;

оценка «удовлетворительно» выставляется обучающемуся, если объем правильного ответ не менее чем 30%;

оценка «неудовлетворительно» выставляется обучающемуся, если объем правильного ответ менее чем 30%.

20.3 Фонд оценочных средств сформированности компетенций студентов, рекомендуемый для проведения диагностических работ

ПК-4 Способен разрабатывать профессионально-ориентированные программные средства и приложения на основе мобильных технологий

ПК-5 Способен выбирать технологии и средства разработки мобильных приложений, определять ключевые сценарии для архитектуры мобильных приложений, разрабатывать новые алгоритмические, методические и технологические решения в сфере разработки мобильных приложений

1)_____ закрытые задания (тестовые, средний уровень сложности):

1. Установите соответствие между типом очереди и описанием:
Главная очередь: Последовательная очередь – Он выполняется в основном потоке.
Глобальная очередь: Параллельная очередь – Он выполняется с разными приоритетами и совместно используется системой intire.
Пользовательская очередь: Последовательная/ параллельная очередь.

2. Установите соответствие между качеством обслуживания (Quality of Service) и типом задач?
User Interactive: работа, выполняемая в основном потоке, например анимация или операции рисования.
User Initiated: Работа, которую начинает пользователь и которая должна дать немедленные результаты. Эта работа должна быть завершена, чтобы пользователь мог продолжить.
Utility: Работа, которая может занять некоторое время и которую не нужно заканчивать сразу. Аналогично индикаторам выполнения и импорту данных.
Background: Эта работа не видна пользователю. Резервное копирование, синхронизация, индексация и т.д.

3. Выберите существующие в Swift уровни управления доступом:
А) private +
Б) private(set) +
В) fileprivate +
Г) private(get)
Д) open +

Е) close

4. Выберите НЕверные утверждения о SwiftUI и UIKit:

А) В UIKit пользовательские интерфейсы создаются с помощью конструктора интерфейсов перетаскивания.
Б) Приложения UIKit подключаются к коду с помощью outlets и actions.
В) В SwiftUI пользовательские интерфейсы создаются программно.
Г) SwiftUI доступен только в iOS 12.0 и более поздних версиях. +

5. Установите соответствие:
А) as - используется для восходящего кастинга
Б) as? - создает значение optional, возвращает значение nil в случае неуспешного кастинга
В) as! - не создает значение optional, создает значение указанного типа, программа завершается с ошибкой в случае неуспешного кастинга

6. Что такое Safe Area?

А) Safe Area позволяет создавать специальные ограничения, чтобы контент не был скрыт специальными аппаратными панелями iOS. +
Б) Safe Area - место, где можно писать потокобезопасный код.
В) Функции, которые не выбрасывают (throw) ошибок

7. Установите соответствие
Self - относится к любому типу, соответствующему протоколу.
self - относится к любому значению, которое содержит тип.

8. Выберите верные утверждения о mutating-функциях в Swift?
А) В Swift свойства типов значений по умолчанию могут быть изменены в его методах экземпляра.
Б) Нужно использовать ключевое слово mutating в методе экземпляра, чтобы изменить свойства типа значения. +
В) Mutating-функция имеет право изменять значения свойств. +
Г) Нужно использовать ключевое слово mutating в методе экземпляра, чтобы преобразовать value-тип в reference-тип.

9. Выберите верные утверждения о цепочках optional в Swift?

А) С помощью цепочек можно связать несколько запросов вместе. +
Б) Если какое-либо звено в цепочке равно нулю, то будет ошибка.
В) Optional цепочка возвращает значение, если вся цепочка завершается успешно. +

2) открытые задания (тестовые, повышенный уровень сложности):

1. Для чего используются контейнеры HStacks и VStacks?

Ответ: контейнеры позиционируют View внутри себя, распределяя пространство между своими сабвью. HStack располагает View горизонально, VStack - вертикально.

2. Перечислите основные отличия декларативной и императивной моделей проектирования пользовательского интерфейса.

Ответ: Императивный синтаксис использует операторы, которые изменяют состояние программы. Императивная программа состоит из команд, которые должен выполнять компьютер. Он фокусируется на описании того, как работает программа, путем реализации алгоритмов в явных шагах. С другой стороны, декларативный синтаксис использует желаемые результаты без явного

перечисления команд, которые должны быть выполнены. Она фокусируется на том, что, а не на том, как. В императивном синтаксисе вам необходимо предоставить пошаговые инструкции для выполнения некоторого действия. В декларативном программировании вам нужно только описать действие, а не то, как его выполнить.

3. Для чего используется @State?

Ответ: @State - обертка свойства, которую можно использовать для обозначения состояния View. SwiftUI хранит ее в специальной внутренней памяти вне структуры View. К ней может получить доступ только связанный с ней View. Как только значение свойства @State меняется, SwiftUI перестраивает View для учета изменения состояния.

4. Дайте определение замыкания.

Ответ: Замыкание - это встроенная функция.

5. Что происходит в данном коде?

let s: String? = …
let ss = s ?? “hello”

Ответ: Оператор ?? используется для создания выражения, которое становится значением по умолчанию, если Optional не установлен. Если значение переменной s будет установлено, то в переменную ss запишется значение ассоциированных данных переменной s. Если значение переменной s не будет установлено (будет .none / nil), то в переменную ss запишется значение по умолчанию “hello”.

1)_____ закрытые задания (тестовые, средний уровень сложности):

1. В какой момент у View выполняется функция .onAppear {}?
А) Когда View появляется на экране +
Б) Когда View собирается показаться на экране
В) Когда View уходит с экрана

2. Что является wrappedValue для @State?
А) Привязанное к чему-то значение
Б) Только Value-тип
В) Что угодно +

3. Что является wrappedValue для @ObservedObject?
А) Привязанное к чему-то значение
Б) Что угодно, что реализует ObservableObject протокол +
В) Что угодно

4. Что является wrappedValue для @Binding?
А) Привязанное к чему-то значение +
Б) Только Value-тип
В) Что угодно

5. Что делает @State?
А) хранит wrappedValue: в “куче” (heap); если она изменяется, делает View недействительным +
Б) делает View недействительным, когда wrappedValue посылает objectWillChange.send()
В) выполняет get / set wrappedValue из другого источника. Когда “привязанное” значение изменятся, делает View недействительным

6. Что делает @ObservedObject?
А) хранит wrappedValue: в “куче” (heap); если она изменяется, делает View недействительным
Б) делает View недействительным, когда wrappedValue посылает objectWillChange.send() +

В) выполняет get / set wrappedValue из другого источника. Когда
“привязанное” значение изменятся, делает View недействительным

7. Что делает @Binding?
А) хранит wrappedValue: в “куче” (heap); если она изменяется, делает View недействительным
Б) делает View недействительным, когда wrappedValue посылает
objectWillChange.send()
В) выполняет get / set wrappedValue из другого источника. Когда
“привязанное” значение изменятся, делает View недействительным +

8. Что является projectedValue для @State?
А) Binding (к тому значению в “куче”) +
Б) Binding (к vars wrappedValue (ViewModel))
В) Binding (self, то есть сам Binding)

9. Что является projectedValue для @State?
А) Binding (к тому значению в “куче”)
Б) Binding (к vars wrappedValue (ViewModel)) +
В) Binding (self, то есть сам Binding)

10. Что является projectedValue для @State?
А) Binding (к тому значению в “куче”)
Б) Binding (к vars wrappedValue (ViewModel))
В) Binding (self, то есть сам Binding) +

11. Что является wrappedValue для @EnvironmentObject?
А) Привязанное к чему-то значение
Б) Только Value-тип
В) ObservableObject, полученный через .environmentObject(), посланный
View +

12. Что делает @EnvironmentObject?
А) хранит wrappedValue: в “куче” (heap); если она изменяется, делает View недействительным
Б) делает недействительным View, когда wrappedValue выполняет
objectWillChange.send() +
В) выполняет get / set wrappedValue из другого источника. Когда
“привязанное” значение изменятся, делает View недействительным

13. Что является projectedValue для @EnvironmentObject?
А) Binding (к тому значению в “куче”)
Б) Binding (к vars wrappedValue (ViewModel)) +
В) Binding (self, то есть сам Binding)

14. Что является wrappedValue для @Environment?
А) Привязанное к чему-то значение
Б) значение некоторой переменной var в EnvironmentValues +
В) ObservableObject, полученный через .environmentObject(), посланный
View

15. Что делает @Environment?
А) хранит wrappedValue: в “куче” (heap); если она изменяется, делает View недействительным
Б) делает недействительным View, когда wrappedValue выполняет
objectWillChange.send()

В) получает (get) / устанавливает (set) значение некоторой переменной var
в EnvironmentValues +

16. Что является projectedValue для @Environment?
А) Binding (к тому значению в “куче”)
Б) Binding (к vars wrappedValue (ViewModel))
В) нет+

2) открытые задания (тестовые, повышенный уровень сложности):

1. Что делает init() в Swift?

Ответ: Метод init() используется для инициализации экземпляра.

2. Что используется для обозначения состояния View?

Ответ: @State

3. В SwiftUI рендер view происходит целиком или блоками?

Ответ: целиком

4. Что позволяет изменить внешний вид View?

Ответ: функции-модификаторы

5. Каким ключевым словом помечены некоторые функции, которые при вызове могут “выбросить” ошибку?

Ответ: throws

6. Как называется протокол, содержащий функции, которые можно использовать для создание некоторой вещи JSON-представляемой?

Ответ: Codable

7. С помощью какого ключевого слова объявляется расширение?

Ответ: extension

8. Что такое декларативный синтаксис?

Ответ: Декларативный синтаксис - это современная языковая парадигма, которая помогает разработчикам писать код процедурно. Используя декларативный синтаксис, разработчик описывает код, который он хочет написать, не беспокоясь о том, как он будет показан / реализован. SwiftUI использует декларативный синтаксис, поэтому вы можете просто указать, что должен делать ваш пользовательский интерфейс. Например: вы можете написать, что вам нужно изображение в нижней части экрана.