РЕСПУБЛИКА ДАГЕСТАН
МБОУ « ГИМНАЗИЯ ГОРОДА БУЙНАКСКА»
368220 г. Буйнакск, ул. Ленина, 42 тел. 2-61-55, 2-22-54

РАССМОТРЕНО

Протокол заседания ШМО учителей_____

от 27.08.2021 года № 1

_____

подпись руководителя МО

СОГЛАСОВАНО

Заместитель директора по УВР

_____ _____

подпись

« 27 » 08 2021г.

УТВЕРЖДЕНО

решением педагогического совета протокол № 1

от _____ 20 года

директор гимназии

Закарьяева А.З._____

ДОПОЛНИТЕЛЬНАЯ ОБЩЕОБРАЗОВАТЕЛЬНАЯ (ОБЩЕРАЗВИВАЮЩАЯ) ПРОГРАММА

технической направленности

«Мобильная разработка»

Возраст учащихся: **10-18 лет** Срок реализации: **1 год**

Педагог дополнительного образования: **Алибекова Рукият Ахмедовна**

Буйнакск, 2021 год

I. ОСНОВНЫЕ ХАРАКТЕРИСТИКИ ПРОГРАММЫ

1. Пояснительная записка

На сегодня разработка программного обеспечения является наиболее востребованным направлением в любых сферах применения. Кроме того, большое развитие мобильных платформ даёт более широкий выбор направлений разработки.

В современном мире Java как платформа является наиболее популярной в связи с тем, что не имеет требований к операционной системе для запуска своих приложений. Кроме того, мобильные устройства на самой популярной ОС Android в большинстве случаев используют приложения, написанные именно на этой платформе. Изучение языка программирования Java по данной программе обучения даёт возможность пользователю мобильного устройства с ОС Android создавать программы в среде разработки, взаимодействующие с элементами графики, аудио и видеофайлами, тестовыми форматами.

Программа «Мобильная разработка» была составлена при поддержке © Samsung R&D Institute Rus (Исследовательский центр Samsung) и при участии Московского физико-технического института (МФТИ).

Проект «IT-школа Samsung» компании Samsung Electronics – это долгосрочная инициатива, которая реализуется при поддержке Министерства образования и науки РФ. В её рамках запланировано бесплатное обучение более пяти тысяч школьников в 20 регионах России по программе дополнительного образования технической направленности в области IT и программирования на языке Java.

Направленность

Программа «Мобильная разработка» имеет техническую направленность, ориентирована на развитие навыков программирования и проектирования программ под платформу Android.

Новизна. **Актуальность**. Данная программа является единственным в своём роде экспериментом в связи с востребованностью на рынке и отсутствием программ образования в данном направлении для школьников. Особенность программы «Мобильная разработка» – в изучении основ языка программирования Java и структуры приложения под ОС Android. Она строится в доступной и понятной для учащихся среде, т.е. программирование ведётся в текстово-графическом режиме, что позволяет сразу задавать необходимый функционал для элементной базы приложения.

Педагогическая целесообразность программы

Программа «Мобильная разработка» составлена в виде модулей, позволяющих получить детям необходимый объём знаний в зависимости от уровня подготовки и потребности.

Стартовый уровень предполагает использование и реализацию общедоступных и универсальных форм организации материала, минимальную сложность предлагаемого для освоения содержания программы.

Базовый уровень предполагает использование и реализацию таких форм организации материала, которые допускают освоение специализированных знаний и

языка, гарантированно обеспечивают трансляцию общей и целостной картины в рамках содержательно-тематического направления программы.

Осваивая данную программу, учащиеся будут овладевать навыками востребованных уже в ближайшие десятилетия специальностей, многие из которых включены в Атлас профессий будущего. Знания, рассматриваемые в программе, будут полезны для каждой перспективной профессии.

Отличительная особенность

Дополнительная общеразвивающая программа «Мобильная разработка» является модульной.

Модуль – структурная единица образовательной программы, имеющая определённую логическую завершённость по отношению к результатам обучения. *(Словарь рабочих терминов по предпрофильной подготовке)*. Каждый модуль состоит из кейсов (не менее двух), направленных на формирование определённых компетенций (hard и soft). Результатом каждого кейса является «продукт» (групповой, индивидуальный), демонстрирующий сформированность компетенций.

Кейс – история, описывающая реальную ситуацию, которая требует проведения анализа, выработки и принятия обоснованных решений (*Высшая школа экономики*).

Кейс включает в себя набор специально разработанных учебно-методических материалов. Кейсовые «продукты» могут быть самостоятельным проектом по результатам освоения модуля или общего проекта по результатам всей образовательной программы.

Модули и кейсы различаются по сложности и реализуются по принципу «от простого к сложному».

Для возрастной категории 10-18 лет при решении кейсов ставятся задания повышенного уровня сложности и применяется оборудование для соответствующей возрастной категории.

Адресат общеразвивающей программы

Дополнительная общеразвивающая программа «Мобильная разработка» предназначена для детей в возрасте 10-18 лет, без ограничений возможностей здоровья.

Группы формируются по возрасту.

Объём общеразвивающей программы составляет 144 часа в год.

Режим занятий, объём общеразвивающей программы:

- 124 часа учебных, занятия проводятся: I полугодие – 2 раза в неделю по 2 академических часа, II полугодие – 2 раза в неделю по 2 академических часа;
- 8 часов отводится на консультации по темам индивидуального проекта (дата и время определяются по согласованию с обучающимися); 2 часа отводится на итоговое занятие (защита).

Формы обучения и виды занятий

Учебный процесс строится таким образом, чтобы экспериментальная и практическая работа преобладала над теоретической подготовкой. Необходимые для работы теоретические сведения находятся на каждом персональном компьютере в специальной папке, даются педагогом перед началом практических занятий. Индивидуальная работа проводится во время практических занятий – при выполнении задания у каждого учащегося возникают свои вопросы. Групповая работа проводится во время теоретических занятий. Каждая тема по программированию сопровождается наглядной демонстрацией работы алгоритма для того, чтобы учащиеся представляли

работоспособность алгоритма, а также к чему им нужно стремиться при выполнении поставленной задачи. Учебный процесс организуется на основе постепенного усложнения учебного материала, как теоретического, так и практического.

Программой предусмотрены следующие виды деятельности обучающихся:

- освоение теоретического и практического материала на занятиях;
- разработка индивидуального проекта;
- участие в вебинарах;
- промежуточная аттестация в форме электронного тестирования;
- самостоятельная практическая работа: выполнение домашних заданий, мини-проектов (небольшие приложения, которые реализуются учениками преимущественно на занятиях совместно с учителем с небольшими самостоятельными доработками в качестве домашнего задания).

По одному из вариантов тестов по каждому модулю представлены в приложениях 6, 7, 8, 9, 10.

По типу организации взаимодействия педагогов с обучающимися при реализации программы используются личностно-ориентированные технологии, технологии сотрудничества.

Реализация программы предполагает использование здоровьесберегающих технологий.

Здоровьесберегающая деятельность реализуется:

- через создание безопасных материально-технических условий;
- включением в занятие динамических пауз, периодической смены деятельности обучающихся;
- контролем соблюдения обучающимися правил работы на ПК;
- через создание благоприятного психологического климата в учебной группе в целом.

Антикоррупционное просвещение обучающихся

Основной мерой по профилактике коррупции является формирование в обществе нетерпимости к коррупционному поведению (статья 6 ФЗ № 273-ФЗ «О противодействии коррупции»). Её реализация связана с повышением уровня правовой культуры, что достигается осуществлением правового воспитания, мероприятиями по антикоррупционному просвещению участников образовательных отношений (обучающихся, родителей, законных представителей несовершеннолетних обучающихся), основанных на знаниях общих прав и обязанностей и направленных на формирование антикоррупционного мировоззрения.

Антикоррупционная направленность правового воспитания основана на повышении в обществе в целом позитивного отношения к праву и его соблюдению; повышении уровня правовых знаний, в том числе о коррупционных формах поведения и мерах по их предотвращению; формировании у государственных, муниципальных служащих и у граждан представления о мерах юридической ответственности, которые могут применяться в случае совершения коррупционных правонарушений. К задачам антикоррупционного воспитания и пропаганды относятся ознакомление граждан с сутью, причинами, последствиями коррупции, поощрение нетерпимости к её проявлениям, демонстрирование возможности борьбы с коррупцией.

Целью антикоррупционного воспитания является воспитание ценностных установок и развитие способностей, необходимых для формирования у молодых людей гражданской позиции в отношении коррупции, негативного отношения к

коррупционным проявлениям. Основной результат антикоррупционного воспитания – подготовка человека, способного выполнять властные полномочия или взаимодействовать с представителями властных структур на правовой основе, избегая подкупа, взяточничества и других неправомерных действий.

2. Цель и задачи общеразвивающей программы

Цель программы – формирование технической грамотности средствами приобщения обучающихся к разработке программ под современную платформу Android.

Задачи

Образовательные:

- расширение знаний о современных и популярных платформах;
- обучение языку программирования Java, языку разметки XML;
- обучение объектно-ориентированному подходу в проектировании и разработке программного обеспечения;
- знакомство с архитектурой приложения под Android;
- обучение программированию технических устройств.

Развивающие:

- формирование алгоритмического мышления; навыков работы с информацией; умения самостоятельно решать поставленную задачу, излагать мысли в чёткой логической последовательности, отстаивать свою точку зрения, анализировать ситуацию и самостоятельно находить ответы на вопросы путём логических рассуждений;
- развитие логического и технического мышления.

Воспитательные:

- воспитание этики групповой работы и отношений делового сотрудничества;
- создание условий для развития устойчивой потребности в самообразовании.

Содержание общеразвивающей программы

Учебный план (по модулям)

№ п/п	Наименование раздела, темы	Количество часов			Формы аттестации / контроля
		Теоретические занятия	Практические занятия	Всего	
	Стартовый уровень				
1.	**Модуль 1. Основы программирования**	**9**	**13**	**22**	
1.1	Среда разработки	1	1	2	
1.2	Примитивные типы данных. Арифметика	1	1	2	
1.3	Операции отношения и логические операции	1	1	2	
1.4	Условные конструкции. Блоки	1	1	2	
1.5	Итеративные конструкции	2	2	4	
1.6	Методы (функции). Видимость переменных	2	2	4	
1.7	Многомерные и неровные массивы	1	1	2	
1.8	Практикум		4	4	
1.9	Контрольное тестирование по модулю		2	2	Тест № 1
2.	**Модуль 2. Объектно-ориентированное программирование**	**10**	**14**	**24**	
2.1	Классы и объекты	1	1	2	
2.2	Классы: конструкторы, статические методы	2	2	4	

2.3	Начальные приёмы тестирования и отладки	1	1	2	
2.4	Архитектура приложений под Android. Активности	1	1	2	
2.5	Интерфейс пользователя. Язык разметки XML	2	2	4	
2.6	Наследование. Намерения	2	2	4	
2.7	Полиморфизм	1	1	2	
2.8	Практикум		4	4	
2.9	Контрольное тестирование по модулю		2	2	Тест № 2
	Базовый уровень				
3.	**Модуль 3. Основы программирования Android-приложений**	**8**	**14**	**22**	
3.1	Практикум ООП проектирования	2	2	4	
3.2	Ввод, вывод и исключения	1	1	2	
3.3	Внутренние классы в обработке событий	2	2	4	
3.4	Параллелизм и синхронизация. Потоки	1	1	2	
3.5	Двумерная графика в Android-приложениях	1	1	2	
3.6	Разработка игровых приложений. Реализация графики на основе SurfaceView	1	3	4	
3.7	Практикум		4	4	
3.8	Контрольное тестирование по модулю		2	2	Тест № 3
4.	**Модуль 4. Алгоритмы и структуры данных**	**13**	**17**	**30**	
4.1	Массивы. Алгоритм двоичного поиска	2	2	4	
4.2	Списки	2	2	4	

4.3	Адаптеры в Android	1	1	2	
4.4	Деревья	1	1	2	
4.5	Рекурсия	1	1	2	
4.6	Алгоритмы сортировки	1	1	2	
4.7	Хэш-таблица и функция хэширования	1	1	2	
4.8	Ассоциативные массивы	1	1	2	
4.9	Реляционная модель данных. СУБД	1	1	2	
4.10	Локальные СУБД. Введение в SQL	2	2	4	
4.11	Практикум		4	4	
4.12	Контрольное тестирование по модулю		2	2	Тест № 4
5.	**Модуль 5. Основы разработки серверной части мобильных приложений**	**9**	**17**	**26**	
5.1	IP-сети	1	1	2	
5.2	Web-сервер. HTTP-запросы и ответы	2	2	4	
5.3	Клиент-серверная архитектура мобильных приложений	2	2	4	
5.4	Облачные платформы. REST-взаимодействие	2	2	4	
5.5	Серверные СУБД	2	4	6	
5.6	Практикум		6	6	
5.7	Контрольное тестирование по модулю		2	2	Тест № 5
6.0	**Итоговая защита**		**2**	**2**	
7.0	**Консультации по ИП**		**8**	**8**	
	Итого:	**49**	**95**	**144**	

Содержание учебно-тематического плана
Стартовый уровень

Модуль 1. Основы программирования

Тема 1.1. 2 часа. Среда разработки.
Тема 1.2. 2 часа. Примитивные типы данных. Арифметика.
Тема 1.3. 2 часа. Операции отношения и логические операции.
Тема 1.4. 2 часа. Условные конструкции. Блоки.
Тема 1.5. 4 часа. Итеративные конструкции.
Тема 1.6. 4 часа. Методы (функции). Видимость переменных.
Тема 1.7. 2 часа. Многомерные и неровные массивы.
Практикум. 4 часа.
Контрольное тестирование по модулю. 2 часа.

Тема 1.1. Среда разработки

Цель. Закрепление на практике базовых навыков работы с Java-программой и основ синтаксиса Java.

Необходимые знания. Представление о том, что такое программа и как она выполняется на компьютере, навыки пользователя ПК.

Среда разработки IntelliJ IDEA/Eclipse. Шаблон программы на Java с функцией main(). О среде разработки IntelliJ IDEA и Eclipse. Понятие проекта. Порядок создания, компиляции, сборки и запуска приложения. Порядок установки среды разработки на домашнем компьютере.

Упражнение 1.1.1. Выполнить первую Java-программу «Здравствуй, мир»:

```
public static void main(String[] args) {
        println(«Hello, user!»);     }
```

На её примере объяснить порядок создания, компиляции и сборки проекта на языке Java, порядок запуска проекта на выполнение.

Разбор Java-проекта. Уяснение факта, что имя класса должно совпадать с именем файла. Объяснение понятий, связанных с ООП, не проводится. Это будет рассмотрено в рамках 2-го модуля программы. На данный момент ученикам предлагается сконцентрироваться на содержимом метода main(). Понятие пакета (package). Роль метода main(). Комментарии. Демонстрация типичных ошибок, возникающих при компиляции и выполнении Java-программ.

Упражнение 1.1.2. Написать и отладить программу, которая выводила бы информацию об ученике в следующем виде: Ф.И.О., школа, класс.

Тема 1.2. Примитивные типы данных. Арифметика

Цель. Рассмотреть примитивные типы данных, арифметические выражения и операторы, операторы присваивания, преобразования типов.

Необходимые знания. Понятия «бит» и «байт»; двоичная, восьмеричная, шестнадцатеричная системы счисления; перевод чисел из одной системы счисления в другую, понятие переменной.

Переменные и **константы**. Данные, обрабатываемые компьютером; описание переменной; задание начального значения переменной; имя переменной. Понятие «тип переменной». Что относят к примитивным типам. Понятие константы. Ключевое слово final.

Целочисленные типы данных. Разновидности типа «целое»: int, short, long, byte. Размер каждого из типов (количество байт), диапазон представления (минимальное и максимальное значения).

Как задать значение константы в десятичной, двоичной, восьмеричной, шестнадцатеричной системе счисления. Как указать, что константа относится к типу long. Вывод на печать данных целого типа. Ввод данных целого типа.

Упражнение 1.2.1. Написание простейших программ, объявляющих переменные целого типа, присваивающих им значения. Вывод этих значений на печать. Наблюдение за поведением компилятора, когда переменной присваивается заведомо некорректное значение или выходящее за пределы диапазона для данного типа.

Программа, манипулирующая представлением целого числа в различных системах счисления. Например, задается число в тексте программы в одной системе счисления, а выводится на печать – в другой.

Типы с плавающей точкой. Типы float, double. Применение суффиксов для указания типа числовых констант.

Упражнение 1.2.2. Написание простейших программ, объявляющих переменные вещественного типа, присваивающих им значения. Вывод этих значений на печать. Наблюдение за поведением компилятора, когда переменной присваивается заведомо некорректное значение или выходящее за пределы диапазона для данного типа.

Арифметические выражения и операторы, операторы присваивания. Сложение, вычитание, умножение, деление, остаток от деления (%), инкремент (++), декремент (--). Префиксная и постфиксная запись инкремента и декремента, уяснение отличия между ними. Операторы присваивания (=, +=, -= и т.д.). Скобки. Рассмотрение подробной таблицы со столбцами: название оператора, форма записи, порядок выполнения. Примеры программ, демонстрирующих применение приоритетов.

Данные типа char. Дать общее представление о кодировке Unicode и UTF-16. Дать рекомендацию по возможности пользоваться не символами, а символьными строками.

Упражнение 1.2.3. Написание простейших программ, демонстрирующих выполнение каждого изученного оператора. Объяснение результатов её выполнения.

Задание 1.2.1. Задачи на определение по значению числа, к какому целочисленному типу оно относится. К какому типу относятся целочисленные константы: 120, 21L, 015, 0b101, 1_000_000, 0x84? Чему равны значения этих констант в десятичной системе счисления?

Задание 1.2.2. Пусть a=7, b=5, c=4. Какие значения будут иметь эти переменные в результате выполнения последовательности операторов? Подсчитать «вручную» и затем проверить результат, написав и запустив программу:

- c=a*b; b=b+1; a=c-b; b=(b+c)/2; c=++b;
- c=a%b; b=a; a*=b; c=a*(b-3); a+=c+b;
- a=b*c; b%=c;b=(a+c)%2; a+=b; c=a*b.

Задание 1.2.3. Определить приоритет операторов и порядок их выполнения:

- 5*6-8/2/2;
- 4-3*3/10;
- 19%6/2*7;
- 1<<2*3+5^4 (в таких выражениях лучше ставить скобки);
- a=5; b=2; a=b*a++.

Тема 1.3. Операции отношения и логические операции

Цель. Рассмотреть операторы и их классификацию, поразрядные операции, логические выражения.

Тип данных boolean. Логические значения true и false. Несовместимость типа boolean с int. Отметить, что приведение логических значений к целым и наоборот невозможно.

Логические операции и операции отношения. Операторы отношения: >, <, >=, <=, !=, ==. Уяснение понятия значения операции отношения как ИСТИННО или ЛОЖНО. Логические операции: логическое И, логическое ИЛИ, логическое НЕ. Тернарная операция ?:.

Выражения и **операции**. По итогу изучения различных операций рассмотрение понятия выражения в языке программирования; знаки операций; знаки-разделители. Классификация операций по количеству операндов: унарные и бинарные. Классификация операций по типу: арифметические, логические, присваивания, отношения и др.

Упражнение 1.3.1. Программа, демонстрирующая выполнение логических операций и операций отношений. Объяснение результатов её выполнения.

Задание 1.3.1. Задачи на «ручное» написание логических выражений средствами языка Java:

- x лежит вне отрезка [a, b];
- x принадлежит отрезку [a, b] или отрезку [c, d];
- x лежит вне отрезков [a, b] и [c, d];
- целое a является нечётным числом;
- целое a является трёхзначным числом, кратным пяти;
- из чисел a, b, c меньшим является c, а большим b;
- среди чисел a, b, c, d есть взаимно противоположные;
- среди целых чисел a, b и c есть хотя бы два чётных;
- из отрезков с длинами a, b, c можно построить треугольник;
- год, заданный числом a, является високосным;
- год, заданный числом a, не является високосным;
- число a является простым;
- среди целых чисел a, b, c есть хотя бы два нечётных;
- отрезки длиной a, b и c могут образовать прямоугольный треугольник.

Тема 1.4. Условные конструкции. Блоки

Цель. Изучить внутреннюю логику работы условных конструкций; приобрести навыки их использования в различных формах, предусмотренных синтаксисом языка. Закрепить навыки написания всех ранее изученных операторов путём написания и вычисления выражений.

Необходимые знания. Здесь и далее необходимы знания в объёме предыдущих тем.

Область действия блоков. Фигурные скобки для выделения блока. Вложенность блоков. На данный момент рассмотреть только ограничение на объявление переменных с одинаковым именем в одном и том же или вложенных блоках.

Конструкция if-else.Синтаксис оператора:

```
if (cond_expression) TRUE_statement;
```

или

```
if (cond_expression) TRUE_statement;
else FALSE_statement;
```

Разъяснить, что statement – это только один оператор или блок. Фундаментальное правило для сложных ветвлений, реализуемых с помощью вложенных конструкций if-else: else относится к ближайшему if, не имеющему else.

Конструкция switch-case. Синтаксис. Что может быть в качестве метки case.

Мотивировка использования конструкции как упрощение сложных ветвлений. Логика выполнения, объяснение роли ключевых слов break и default в конструкции switch-case.

Упражнение 1.4.1. Небольшие фрагменты кода, иллюстрирующие использование операторов ветвления, приоритетов вычисления операторов в выражении. Ускоренное вычисление логических выражений – прекращение вычислений, когда результат уже ясен.

Задание 1.4.1. Написать собственный пример на использование операторов ветвления. Например, нахождение максимума, минимума среди нескольких введённых переменных.

Подготовка заданий. При подготовке заданий на написание учеником программ можно использовать автоматическую систему тестирования (например, http://informatics.mccme.ru). Так же можно подготовить задачу самостоятельно с использованием модульного тестирования.

Рассмотрим пример тестов для задачи минимума трёх чисел:

```
public void testEasy() {
        runTest("4 5 7\n", "4");
        runTest("5 7 3\n", "3");
        runTest("4 1 7\n", "1");
}
```

Можно добавить много тестов, сгенерированных циклом:

```
public void testEasy() {
        for (int i = 0; i < 30; ++i) {
                for (int j = i; j < 30; ++j) {
                        for (int t = j; t < 30; ++t) {
                                runTest("" + i + " " + j + " " + t " +\n", "" + i);
                                runTest("" + j + " " + i + " " + t " +\n", "" + i);
                                runTest("" + j + " " + t + " " + i " +\n", "" + i);
}       }       }       }
```

Тема 1.5. Итеративные конструкции

Цель. Изучить внутреннюю логику работы итеративных конструкций; приобрести навыки их использования в различных формах, предусмотренных синтаксисом языка. Изучить оператор for, for each, одномерные массивы.

Цикл с предусловием while. Синтаксис. Объяснение логики работы, пример использования.

Цикл с постусловием do-while. Синтаксис. Объяснение логики работы, пример использования. Уяснение ключевого отличия от цикла while с предусловием: цикл с постусловием выполняется хотя бы один раз.

Операторы прерывания логики управления программой. Безусловные операторы перехода break, continue.

Упражнение 1.5.1. Небольшие фрагменты кода, иллюстрирующие использование операторов цикла (без использования массивов). Например, вычисление НОД по алгоритму

Евклида.

Задание 1.5.1. Написать собственный пример на использование операторов цикла и операторов безусловного перехода. Например, проверка числа на то, что оно является простым.

Массивы. Определение массива как совокупности элементов одного и того же типа, расположенных вплотную друг за другом в памяти. Объявление массива двумя способами. Подчеркнуть необходимость создания массива с помощью new(). Значения, инициализируемые массивом по умолчанию.

Инициализация массива без new() – инициализация массива при объявлении. Доступ к отдельным элементам массива. Определение количества элементов в массиве через свойство length.

Цикл for. Синтаксис. Логика работы, роль каждой из составных частей. Частные формы записи оператора for: отсутствует инкрементальное выражение; отсутствует инкрементальное выражение и начальное выражение. Уяснение связи между for и while, эквивалентная запись for через while. Примеры некорректного использования операторов цикла, приводящего к зацикливанию. Вложенные циклы for.

Цикл for each. Синтаксис. Преимущества его применения при работе с массивами в сравнении с обычным for. Отметить, что переменная в цикле for each перебирает не индексы массива, а сами элементы массива.

Упражнение 1.5.2. Фрагменты кода, иллюстрирующие на одномерном массиве решение задач нахождения максимального, минимального.

Задание 1.5.2. Написать программу по обработке массива с выводом на экран полученного результата:

- поиск заданного элемента простым перебором;
- переворот массива «задом наперёд» без использования вспомогательного массива;
- вычисление суммы элементов массива;
- нахождение самого часто повторяющегося числа среди элементов массива;
- нахождение среднего арифметического числа элементов массива;
- заполнить массив числами арифметической прогрессии по заданной учителем формуле.

Тема 1.6. Методы (функции). Видимость переменных

Цель. Усвоить фундаментальное понятие функции в программировании и проектировании программного обеспечения на примере методов Java; приобрести навыки их использования. Рассмотреть видимость переменных.

Основные понятия. Определение функции как логически самостоятельной именованной части программы, которой могут передаваться параметры, и которая может возвращать какое-то значение.

Определение функции. На примере объяснить понятия: тело метода, тип возвращаемого значения. Список формальных аргументов, список фактических аргументов. Методы с типом void и методы с пустым списком аргументов.

Упражнение 1.6.1. На примере продемонстрировать ситуации, когда функции необходимы. Привести в качестве примера функции println и readInt.

Реализовать собственную функцию для считывания и вывода массива (int[] readIntArray(int length) и void printArray(int[] a, char delimiter)) с использованием уже

существующих функций.

Область видимости переменных. Обзорная классификация переменных по области видимости: область класса, область метода, область блока.

Тема 1.7. Многомерные и неровные массивы

Цель. Изучить многомерные массивы на примере двумерных.

Матрицы. Отдельно отметить, что массивы в Java имеют особенность: в языке на самом деле нет многомерных массивов, а только одномерные, т.е. многомерные массивы - это искусственно создаваемые “массивы массивов”. Разбор примеров с матрицами: объявление, инициализация, обращение к элементам. Использование вложенного цикла for each.

Упражнение 1.7.1. Фрагменты кода, иллюстрирующие обработку матриц. Например, заполнение матрицы случайными числами по заданным правилам, поиск или обработка элементов в указанных столбцах, строках и диагоналях.

Неровные массивы. Проиллюстрировать, как массивы второго уровня могут иметь различную размерность. Как результат – получаем неровные массивы.

Упражнение 1.7.2. Фрагменты кода, иллюстрирующие особенности объявления и обработки неровных массивов.

Задание 1.7.1. Написать программу поиска максимального/минимального числа, заданного числа:

- на диагоналях матрицы;
- в чётных или нечётных строках;
- в чётных или нечётных столбцах.

Задание 1.7.2. Написать программу генерации игрового поля для головоломки Судоку размером 9x9. Написать программу решения заданной головоломки Судоку размером 9x9.

Практикум

Практическое занятие по темам модуля. Конкретное содержание определяется учителем.

Контрольное тестирование по модулю

Электронный тест охватывает все темы и в большей части направлен на оценку практических знаний и навыков учеников.

Модуль 2. Объектно-ориентированное программирование

Тема 2.1. 2 часа. Классы и объекты.
Тема 2.2. 4 часа. Классы: конструкторы, статические методы.
Тема 2.3. 2 часа. Начальные приемы тестирования и отладки.
Тема 2.4. 2 часа. Архитектура приложений под Андроид. Активности.
Тема 2.5. 4 часа. Интерфейс пользователя. Язык разметки XML.
Тема 2.6. 4 часа. Наследование. Намерения.
Тема 2.7. 2 часа. Полиморфизм.
Практикум. 4 часа.
Контрольное тестирование по модулю. 2 часа.

Тема 2.1. Классы и объекты

Цель. Уяснить различие между процедурным (структурным) и объектным подходом к программированию; освоить понятия «класс», «объект».

Основные понятия. Взгляд на ООП как более естественное по сравнению с процедурным подходом отражение в программировании реалий окружающего мира и отношений между ними. Интуитивно-понятное объяснение понятий: класс, объект (экземпляр класса), переменные (поля) класса, метод класса. Иллюстрация на примерах окружающего мира и примерах школьной математики. Сопоставление каждому понятию некоего утверждения, афористично и емко раскрывающего его суть:

Объект – «всё есть объект», класс – «каждый объект имеет тип», переменные – «каждый объект имеет собственную память, отличную от других объектов», метод – «все объекты определенного типа могут принимать одинаковые сообщения», программа на ОО-языке – «связка объектов, говорящих друг другу что делать, посылаю сообщения или, иными словами, вызывая методы».

Описание протокола класса. Ключевое слово class как начало описания нового типа данных.

Описание полей класса. Метод класса, его аргументы и возвращаемое значение. Описание метода в протоколе класса.

Создание объекта класса с помощью оператора new. Обращение к полям класса через.

Вызов метода через переменную – объект собственного класса. Интерпретация метода как посылки сообщения объекту.

Инкапсуляция, **наследование**, **полиморфизм**. Общее понятие парадигм ООП инкапсуляция, полиморфизм и наследование на примерах из жизни.

Более подробно остановиться на инкапсуляции: уяснение целесообразности скрытия внутреннего строения объекта и ограничения доступа к его полям – взаимодействие с объектом организуется только через его методы. Взгляд на инкапсуляцию как на средство защиты целостности данных объекта: объект «Интервал» (левая граница не должна стать больше правой).

Упражнение 2.1.1. Проектирование и реализация простейшего класса, описывающего рациональную дробь. Описание полей (числитель, знаменатель). Объяснение, почему эти поля должны быть закрытыми (исключение деления на ноль). Автоматическая генерация getter/setter в классе с помощью среды разработки.

Обзор классов, соответствующих примитивным типам. Знакомство с классами Integer, Character, Float и т. д.

Упражнение 2.1.2. Разбор примера с применением классов Integer, Character, Float и т. д.

Упражнение 2.1.3. Проектирование классов и их взаимодействия для проведения урока. Например, Учитель готовит Задание, Ученик выполняет Задание и получает Решение, Учитель проверяет Решение и ставит Оценку и т. п.

Задание 2.1.1. Придумать примеры классов и соответствующие им примеры объектов, полей и методов.

Тема 2.2. Классы: конструкторы, деструкторы и статические методы

Цель. Познакомиться с примерами java-кода, описывающего классы, приобрести первый опыт проектирования и реализации полноценного, логически завершённого класса, освоить понятие перегрузки методов, способы инициализации данных в программе на Java.

Конструкторы и **деструкторы**. Конструкторы и деструкторы в Java и их использование. Разновидности конструкторов: без аргументов; с аргументами. Понятие конструктора по умолчанию. Особенности автоматической генерации конструктора по умолчанию.

Перегрузка методов. Уяснение возможности иметь несколько методов с одним и тем же именем, но разными сигнатурами (наборами аргументов) на примере конструктора класса. Распространение концепции перегрузки на любой метод Java. Пример перегрузки конструктора и обычного метода. Необходимость уникального списка типов аргументов для различения перегруженных методов. Уникальность как совокупность количества аргументов, их типов и порядка следования. По типу возвращаемого значения метод не может быть перегружен!

Ключевое слово this. Уяснение смысла ключевого слова this как ссылки на текущий объект. Примеры типового использования: возврат в return ссылки на текущий объект; различение имени локальной переменной и имени поля класса при их совпадении.

Спецификаторы доступа. Ключевое слово public и интерфейсный доступ к объектам класса. Ключевое слово private для описания закрытых полей и методов класса. Понятие доступа класса. Ключевое слово public по отношению к классу (а не члену класса) и его смысл.

Упражнение 2.2.1. Продолжение разработки класса, описывающего рациональную дробь. Реализация конструкторов. Реализация методов экземпляра: модификация числителя; модификация знаменателя; проверка дроби на правильность; выделение целой части; выделение дробной части; вывод дроби на печать; результат деления в виде десятичной дроби; сложение с дробью; умножение на дробь; деление на дробь; вычитание дроби.

Статические методы класса. Ключевое слово static и статические члены класса. Интерпретация статических методов как методов класса в отличие от остальных методов – методов экземпляра. Обращение к статическим членам класса через имя класса (а не переменную – объект).

Инициализация различных **типов данных**. Инициализация полей класса в конструкторе. Явная и неявная инициализация примитивных типов и объектных ссылок. Инициализация статических данных. Инициализация массива, примеры.

Упражнение 2.2.2. Разбор примера: класс для демонстрации значений типов по умолчанию, показать, что поля класса инициализируются еще до вызова конструктора, в каком бы порядке не стояли в классе описания полей и тело конструктора. Разбор примера демонстрации создания и инициализации многомерных массивов.

Задание 2.2.1. Написать функцию run(), тестирующую класс «Рациональная дробь». Функция должна создавать экземпляры класса, выполнять реализованные в классе методы и выводить результат. Реализация статических методов класса: сложение дробей; вычитание дробей; умножение дробей; деление дробей.

Модифицируйте функцию print, чтобы вывод при необходимости был в виде смешанной дроби, убедитесь в корректности работы с отрицательными числами.

Модифицируйте конструктор дроби, чтобы все хранимые дроби были несократимы.

Тема 2.3. Начальные приемы тестирования и отладки

Цель. Ознакомиться с приёмами, позволяющими найти ошибку в программе и предотвратить ошибки при внесении изменений в программу на примерах со строками.

Отладочный вывод и логирование. Вывод значений переменных на экран для последующего анализа их значений. Понятие логирования и его области применения. Вывод отладочных данных с помощью android.util.Log и просмотр этих логов в (Window | Show View | Other... | Android | LogCat.).

Использование отладчика. Пошаговое выполнение программы и просмотр переменных. В отличие от отладочного вывода не требует изменения программы. Однако не позволяет легко просматривать сложные конструкции (сумма элементов массива, диагональ матрицы).

Breakpoint для перехода на конкретную строчку программы (в том числе с указанием условия остановки).

Использование макроса (функции) assert. Проверка инвариантов в программе с помощью макроса assert. Самопроверка программы при каждом её запуске. Проверка входных параметров функции (на примере sqrt или log). Для включения функций assert необходимо установить параметр -ea в Run Configurations -> Arguments -> VM arguments (по умолчанию assert игнорируются, чтобы не замедлять программу).

Тестирование отдельных функций (модульное тестирование). Реализация отдельной функции, проверяющей правильность работы некоторой части программы. Отличие от assert. Выделение легко тестируемой (например, не читающей из входного потока) части программы. Написание модульного теста с помощью библиотеки JUnit.

Работа со строками. Краткий обзор классов String, StringBuffer, StringBuilder. Обратить внимание на то, что объекты класса String являются неизменяемыми.

Упражнение 2.3.1. Разобрать примеры программ, продемонстрировать изученные приёмы тестирования.

Задание 2.3.1. Написать программу по обработке строк. При отладке использовать отладчик и изученные приёмы тестирования.

Тема 2.4. Архитектура приложений под Андроид. Активности

Цель. Познакомиться со средой разработки Android-приложений, рассмотреть общую структуру Android-приложения; создать первое приложение.

Знакомство со средой разработки приложений под Android. О среде разработки Android Studio. Порядок создания, компиляции, сборки и запуска в среде. Порядок установки IDE и эмулятора для разработки приложений под Android на домашнем компьютере.

Общая структурная **схема** приложения под Android. Разбор и комментирование схемы (используется схема из developer.android.com). Жизненный цикл Android-приложения.

Активности (Activity) и их жизненный цикл в Android. Создание Activity. Жизненный цикл Activity. Стеки Activity. Состояния Activity.

Отслеживание изменений состояния Activity. Класс Activity, методы onCreate, onStart, onPause, onStop, onRestart, onResume, onDestroy.

Упражнение 2.4.1. Разбор кода простейшего Android-приложения, иллюстрирующего общую схему, его запуск. В материалах приведён пример приложения, решающего линейное уравнение.

Задание 2.4.1. Модифицировать приложение так, чтобы оно решало квадратное уравнение, а также корректно обрабатывало ситуацию нулевых коэффициентов.

Тема 2.5. Интерфейс пользователя. Язык разметки XML

Цель. Рассмотреть способы задания расположения элементов управления на экране устройства; уяснение необходимости задания расположения универсально для многих устройств. Изучить общую структуру языка XML: понятие тэга, опций тэга, вложенных тэгов, сокращенных тэгов (без закрытия), комментариев.

Примеры использования XML. XML-формат, одновременно понятный человеку и компьютеру, используется для записи конфигурации программ, для передачи данных по сети, хранения данных и другого.

Привести аналогию языка разметки с описанием полей класса. Использование XML в программировании Android-приложений.

Структура документа и комментарии. Пролог, корневой элемент, остальные элементы и их разметка, секция CDATA. Способ записи комментария в XML-документе.

Упражнение 2.5.1. Разобрать пример задания информации в xml (например, описание рецепта или геометрической фигуры).

Описание ресурсов Android с помощью XML. Описание, назначение файлов res/layout/fragment_main.xml и res/values/strings. Обоснование, для чего используется описание всех строк в отдельном файле (удобный поиск и редактирование всех строк, локализация продукта). Основные элементы интерфейса и их описание в файле fragment_main.xml: EditText, TextView.

Разметки (Layouts). LinearLayout и его ориентации, TableLayout. Описание Layout в xml-файле. Вложенные Layout.

Упражнение 2.5.2. Расположить кнопки в форме буквы П с помощью LinearLayout.

Представления (Views). Примеры представлений: TextView, EditText и ProgressBar. Описание представлений в конфигурационном файле. Поиск представлений по их идентификатору findViewById(R.id.name).

Упражнение 2.5.3. Разобрать работу функции println в классе Program в testbed. Разобрать первые две строки processButton для получения входных и выходных данных в testbed.

Задание 2.5.1. Создать xml, описывающий список вещей, которые ученик носит в школу. Каждой вещи должен соответствовать отдельный тэг с её дополнительными свойствами (цвет, название, автор и т. п.).

Задание 2.5.2. Добавить progress bar на экран и передать экземпляр в конструктор класса Program. Расширить Program приватным методом, устанавливающим прогресс обработки. Воспользуйтесь классом BigInteger для поиска первой цифры числа n! (факториал n). В процессе вычисления обновляйте progressbar.

Тема 2.6. Наследование. Намерения

Цель. Изучить понятие интерфейса, возможности наследования классов и приобрести навыки их использования; уяснить различие между отношениями наследования и вложенности.

Наследование. Наследование классов как создание новых классов на основе уже существующих. Отношение «является» между классами. Расширение базового класса новыми полями и методами в классе наследнике, характерными именно для производного класса. Примеры наследования (человек – студент, четырёхугольник – ромб и пр.). Взгляд на наследование классов как на естественную возможность повторного использования кода и независимого расширения библиотек классов. Интерфейс как задание набора методов всех

классов, его реализующих (задание шаблона). Более подробное использование будет рассмотрено в теме «Полиморфизм».

Инициализация базового класса. Синтаксическое описание наследования классов и реализации интерфейсов, ключевое слово extends и implements. Подобъект базового класса внутри объекта производного класса и необходимость его инициализации. Вызов конструктора базового класса как часть конструктора производного класса. Гибкое манипулирование вызовами конструкторов с помощью ключевого слова super.

Защищенные члены класса. Ключевое слово protected и пример мотивации его использования (приведён в приложении): открытый член класса для доступа из производных классов и закрытый для доступа извне.

Упражнение 2.6.1. Разобрать пример с описанием классов, наследованием, переопределением метода, доступами и т. д.

Приведение к **базовому типу**. Уяснение ключевого правила: объект производного класса является объектом базового класса, но не наоборот. Пример использования этого правила в Java: объект неявно приводится к типу своего родителя. Стоит сообщить об этом, более подробно это будет разобрано в теме «полиморфизм».

Ключевое слово final. Применение final к примитивным типам данных: значение переменной не может быть изменено. Применение final к объектам: объектная ссылка не может быть изменена, но содержимое объекта – может. Применение final к аргументу метода. Применение final к методу: метод не может быть переопределён в производном классе. Применение final к классу: у этого класса не может быть наследников.

Контекст. **Намерения** (Intents). Что такое Context и его использование. Явные и неявные намерения.

Создание нескольких Activity (и экранов) в одном приложении. Регистрация их в AndroidManifest.xml. Создание xml файла с описанием вида экрана в папке res/layout. Метода setContentView для загрузки xml файла из res/layout. Создание Intent для перехода на другой экран (метод startActivity).

Упражнение 2.6.2. Разобрать аналогичный пример, где одно из наследований заменено вложенностью классов. Объяснить различие в реализации.

Задание 2.6.1. В соответствии с заданным вариантом задания разработать собственный класс, аналог которого общеупотребителен в окружающем мире или в математике и имеет хорошо знакомое поведение. Для школьника задание должно быть сформулировано учителем.

Возможные варианты заданий:

Реализовать класс «Билет на общественный транспорт». Реализовать классы-наследники: автобусный билет, билет на метро, единый билет (на несколько видов транспорта одновременно). Должны быть реализованы методы для прохода в автобус и метро.

Реализовать класс «Переводчик» для перевода из десятичной системы счисления в другую. Его наследники: в двоичную, в римскую.

Реализовать класс «Нота». Поля: перечислимый тип для ноты (до, ре, ми, фа, соль, ля, си); перечислимый тип для тональности (контроктава, большая, малая, первая, вторая, третья, четвертая); перечислимый тип для знака альтерации (отсутствует, диез, бемоль). Конструктор проверяет корректность задания ноты – не может быть до-бемоль, фа-бемоль, ми-диез, си-диез. Класс Нота обязательно сделать реализующем интерфейс Comparable и написать метод compareTo для сравнения нот: нота с меньшей высотой предшествует (меньше) ноты с большей высотой. Реализовать класс «Музыкальный интервал» как контейнер, содержащий

две ноты. Конструктор должен проверять, что первая нота ниже (меньше) второй ноты. Метод: вычисление длины интервала в тонах. Реализовать и как метод класса, и как метод экземпляра.

Реализовать иерархию классов: «Шахматная фигура», «Пешка», «Слон», «Конь», «Ладья», «Король», «Ферзь». Для каждого производного класса должна быть реализована своя функция «Сделать ход». Члены класса: поле, на котором стоит Фигура (реализовать приватный класс «Поле»); цвет фигуры. Если ход возможен, поле фигуры меняется. Если невозможен – не меняется. Поле характеризуется буквой (a-h и цифрой 1-8). В конструкторах обеспечить контроль ввода (чтобы не поставить, например, белую пешку на первую горизонталь).

Реализовать иерархию классов: «Шашка», «Простая шашка» (производный класс), «Дамка» (другой производный класс). Для каждого производного класса должна быть реализована своя функция «Сделать ход». Члены класса: поле, на котором стоит Шашка (реализовать приватный класс «Поле»); цвет шашки. Если ход возможен, поле шашки меняется. Если невозможен – не меняется. Поле характеризуется буквой (a-h и цифрой 1-8). В конструкторах обеспечить контроль ввода (чтобы не поставить шашку на белое поле).

Тема 2.7. Полиморфизм

Цель. Освоить понятие полиморфизма и познакомиться с примерами его применения.

Общие сведения. Взгляд на полиморфизм как на отделение интерфейса от реализации. Уяснение того, что по-разному реализованными объектами можно управлять, используя один и тот же интерфейс, независимо от того, как реализованы эти объекты. Понятие позднего связывания и его связь с наследованием. Пример с циклическим вызовом одного и того же метода, например, «Повернуть» в массиве геометрических фигур – для каждой фигуры, в зависимости от того, объектом какого производного класса она является, будет вызвана своя реализация этого метода.

Суть понятия полиморфизма и полиморфного метода: в коде метод вызывается через ссылку на объект базового класса; фактический класс объекта определяется во время выполнения программы – позднее связывание; фактически при выполнении программы вызывается реализация метода именно для того класса, к которому принадлежит объект, а не реализация для базового класса.

Упражнение 2.7.1. Рассмотрение типового примера полиморфизма на основе иерархии геометрических фигур и метода draw () – отрисовать.

Преимущества полиморфизма. Объяснение возможностей независимого расширения иерархии классов, предоставляемых полиморфизмом: разработчику нового класса-наследника не нужно ничего менять в уже написанном коде.

Абстрактные классы и методы. Понятие абстрактного класса и абстрактного метода. Синтаксис и мотивация использования. Улучшение кода примера путём объявления базового класса «Фигура» абстрактным.

Задание 2.7.1. Реализовать классы круг и овал, реализующее интерфейс Figure.

Практикум

Практическое занятие по темам модуля. Конкретное содержание определяется учителем.

Контрольное тестирование по модулю

Электронный тест охватывает все темы модуля и в большей части направлен на оценку практических знаний и навыков учеников, полученных в ходе изучения модуля.

Базовый уровень

Модуль 3. Основы программирования Android приложений

Тема 3.1. 4 часа. Практикум ООП проектирования.

Тема 3.2. 2 часа. Ввод, вывод и исключения.

Тема 3.3. 4 часа. Внутренние классы в обработке событий.

Тема 3.4. 2 часа. Параллелизм и синхронизация. Потоки.

Тема 3.5. 2 часа. Двумерная графика в Android-приложениях.

Тема 3.6. 4 часа. Разработка игровых приложений. Реализация графики на основе SurfaceView.

Практикум. 4 часа.

Контрольное тестирование по модулю. 2 часа.

Тема 3.1. Практикум ООП-проектирования

Цель. Освоить приемы ООП-проектирования Android-приложений, изучение простейших диаграмм UML.

Упражнение 3.1.1. Разбор примера проектирования класса Дробь. Демонстрация преимуществ ОО-подхода.

Принципы SOLID. Принцип единственной обязанности (Single Responsibility Principle). Принцип открытости/закрытости (Open-Closed Principle). Принцип подстановки Барбары Лисков (Liskov Substitution Principle, кратко – LSP). Принцип разделения интерфейса (Interface Segregation Principle). Принцип инверсии зависимостей (Dependency Inversion Principle).

Диаграммы UML. UML как язык графического описания для объектного моделирования в области разработки программного обеспечения. Знакомство с инструментальной средой для создания UML-диаграмм. Диаграмма классов как подмножество диаграмм UML: основные элементы и синтаксис.

Упражнение 3.1.2. Разбор примера проектирования игры-квеста. Демонстрация на примере выделения сущностей, методов и полей классов.

Минипроект 3.1. Самостоятельное проектирование UML-диаграммы классов приложения согласно заданию. Выдача заготовки проекта для дальнейшей реализации мини-проекта.

Тема 3.2. Ввод, вывод и исключения

Цель. Изучить файловый ввод вывод и механизм обработки исключений в Java и освоить понятие «исключение» как объект.

Обработка исключений как средство создания надёжного, помехоустойчивого кода. Основные понятия: исключительная ситуация; обработчик исключения; выбрасывание исключения с аргументом с помощью throw; передача управления из текущего контекста наверх.

Обработка исключения с помощью конструкции try-catch. Возможность соответствия одному блоку try нескольких блоков catch. Основные методы класса Exception. Стратегии обработки исключений: прерывание и возобновление. Пример целесообразности

возобновления.

Класс File и его методы: exists(), renameTo(), getAbsolutePath(), canRead(), canWrite(), getName(), length(), isDirectory(), lastModified(). Примеры возникновения и обработки исключений, возникающих при выполнении методов класса.

Упражнение 3.2.1. Разбор примера кода с типовым использованием потоков ввода/вывода: чтение из файла и из памяти; вывод в файл. Использовать классы Scanner и PrintWriter.

Минипроект 3.1. Продолжение. Реализация обработки исключений в заготовке согласно заданию.

Задание 3.2.1. Реализация посимвольного сравнения двух файлов или страниц в интернете. Выводить требуется все отличающиеся символы в произвольном формате. Если символов очень много нужно вывести только часть и количество различий.

Тема 3.3. Внутренние классы в обработке событий

Цель. Освоить применение внутренних анонимных классов для обработки событий интерфейса Android-приложений.

Внутренние (вложенные) **классы**. Понятие внутреннего класса. Отличие от наследования. Назначение. Доступ к состоянию объекта с помощью внутреннего класса.

Локальные и **анонимные** внутренние классы. Сущность, синтаксис, назначение.

Обработка событий пользовательского интерфейса. Краткий обзор классов и интерфейсов для обработки событий. Классы Listeners. Использование анонимных классов для реализации обработчиков событий.

Упражнение 3.3.1. Разбор примера кода с обработчиками событий.

Минипроект 3.1. Завершение. Реализация обработчиков событий с использованием анонимных классов согласно заданию.

Тема 3.4. Параллелизм и синхронизация. Потоки

Цель. Освоить понятия потока, назначения многопоточности и структуры многопоточной программы, получить базовые навыки реализации многопоточности в Java.

Общие понятия. Введение в естественный параллелизм алгоритмов. Оценка улучшения производительности при параллельном выполнении программы. Пример нарушения целостности данных при неаккуратной реализации параллелизма и необходимость синхронизации параллельных ветвей.

Потоки (threads) как средство реализации параллелизма в рамках одного процесса (программы). Общее описание того, как работает многопоточная программа. Некоторые «подводные камни» реализации многопоточности; возможная неопределенность при определении порядка доступа к общему ресурсу.

Процессы и **потоки** в Android. Реализация и запуск AsyncTask. Альтернативные способы создания потокового класса. Наследование от класса Thread и реализация интерфейса Runnable. Предпочтения использования того или иного способа.

Реализация логики потока. Метод run () как реализация логики потока. Запуск потока на выполнение с помощью метода start () и смысл его аргумента как ссылки на объект класса, чей метод run () должен выполнять данный поток. Уяснение фундаментального факта, что вызов start () является асинхронным.

Синхронизация потоков. Объявление метода с помощью ключевого слова syncronized. Применение synchronized к отдельным блокам кода внутри run (). Методы

взаимодействия потоков: suspend () – resume (), wait () -notify (). Метод join () как команда одному потоку дождаться завершения выполнения другого.

Блокировки. Понятие мёртвой блокировки (deadlock), механизм её возникновения.

Упражнение 3.4.1. Разобрать пример программы, совершающей загрузку картинки из интернета и устанавливающей её на экран.

Тема 3.5. Двумерная графика в Android приложениях

Цель. Получить навыки использования двумерной графики в Android. Если графика уже была изучена в рамках тем первого модуля, то можно следующей теме 3.9. посвятить больше времени.

Класс Canvas. Обзор методов и полей класса. Создание собственного View с методом onDraw. Вызов setContentView с вновь созданным классом. Класс Paint. Закраска экрана фоновым цветом. Рисование на canvas круга, прямоугольника. Отрисовка текста. Поворот полотна canvas.rotate и convas.resotre. Отрисовка текста под наклоном.

Двумерная анимация. Обзор методов и полей класса. Создание собственного View с методом onDraw. Вызов setContentView с вновь созданным классом. Класс Paint. Закраска экрана фоновым цветом. Рисование на canvas круга, прямоугольника. Отрисовка текста. Поворот полотна canvas.rotate и canvas.restore. Отрисовка текста под наклоном. Создание массива абсцисс, ординат и скоростей для движения различных элементов.

Упражнение 3.5.1. Разбор предоставленного кода игрового приложения «Крестики – нолики». Внесение изменений в код, пересборка проекта и просмотр влияния изменений на поведение приложения.

Реализовать пример добавления в программу ещё одного экрана с графикой и описанием правил игры.

Проект состоит из трех java-файлов:

- TicTacToe.java – обработка начального экрана игры;
- Game.java – обработка процесса игры;
- GameView.java – отрисовка игрового поля.

В папке res находятся xml-файлы, описывающие параметры отображения (res/layout), строковые константы (res/values/strings), описание массивов (res/values/arrays).

В файле AndroidManifest.xml описываются основные параметры приложения.

Задание 3.5.1. Реализуйте три экрана:

- считает количество нажатий на кнопку;
- загадывает случайное число, которое нужно угадать. Программа может сообщать больше или меньше введённое число загаданного;
- показывает две дроби и нужно определить, какая из них больше.

На каждом экране сделать возможность перейти на следующий экран. Кнопка назад должна возвращать на предыдущий. В качестве доп. задания можно сохранять информацию между экранами с помощью методов putExtra и getIntent().getExtras().get(KEY).

Тема 3.6. Разработка игровых приложений. Реализация графики на основе SurfaceView

Цель. Получить навыки разработки простейших игровых приложений под Android.

Общие подходы для реализации игровых приложений. Последовательные этапы проектирования и реализации игрового приложения. Профессии в мире индустрии игр.

Понятие игрового движка и его использование при разработке игры.

Класс SurfaceView. Обзор. Отличие View от SurfaceView. Особенность класса SurfaceView – предоставляет отдельную область для рисования, действия с которой должны быть вынесены в отдельный вспомогательный поток приложения. Методы getHolder(), lockCanvas(), unlockCanvasAndPost().

Упражнение 3.6.1. Разбор примера простейшей игры с анимацией.

Практикум

Практическое занятие по темам модуля. Конкретное содержание определяется учителем.

Контрольное тестирование по модулю

Электронный тест охватывает все темы модуля и в большей части направлен на оценку практических знаний и навыков учеников, полученных в ходе изучения модуля.

Модуль 4. Алгоритмы и структуры данных

Тема 4.1. 4 часа. Массивы. Алгоритм двоичного поиска.
Тема 4.2. 4 часа. Списки.
Тема 4.3. 2 часа. Адаптеры в Android.
Тема 4.4. 2 часа. Деревья.
Тема 4.5. 2 часа. Рекурсия.
Тема 4.6. 2 часа. Алгоритмы сортировки.
Тема 4.7. 2 часа. Хэш-таблица и функция хэширования.
Тема 4.8. 2 часа. Ассоциативные массивы.
Тема 4.9. 2 часа. Реляционная модель данных. СУБД.
Тема 4.10. 4 часа. Локальные СУБД. Введение в SQL.
Практикум. 4 часа.
Контрольное тестирование по модулю. 2 часа.

Тема 4.1. Массивы. Алгоритм двоичного поиска

Цель. Изучение массивов на примере библиотечных классов Arrays, ArrayList. Знакомство с идеей, применением и реализацией алгоритма двоичного поиска.

Повторение основных сведений о массивах в Java. Синтаксис объявления массива. Явное возвращение массива в качестве результата функции: здесь следует подчеркнуть принципиальное отличие от C, где массив может быть возвращён только неявным образом – по адресу, переданному в одном из параметров функции.

Библиотечный класс Arrays. Класс java.utils.Arrays как набор статических методов, полезных для работы с массивами. Заполнение массива с помощью метода Arrays.fill(). Копирование массива с помощью метода System.arraycopy(). Сравнение массивов на равенство с помощью метода Arrays.equals(): уяснение того факта, что программа должна знать, как же ей сравнивать отдельные элементы массивов на равенство, в связи с чем необходимо иметь реализованным метод equals() для класса, к которому принадлежат элементы массива, если они не относятся к одному из примитивных типов.

Упражнение 4.1.1. Практическое занятие по библиотечному классу Arrays, реализующему массивы: заполнение, копирование, сравнение, печать, другие общие методы.

Библиотечный класс ArrayList как реализация массива без ограничений на количество элементов. Создание списка, метод add(), метод get(), метод size(). Параметризация списка типом объектов при создании: *ArrayList<Person> list = new ArrayList<Person>()*.

Понятие итератора и его использование для обхода контейнера. Итератор как объект. Фундаментальные методы любого итератора: next() – получить следующий объект контейнера; hasNext() – проверить, если ли ещё объекты в контейнере, т.е. не завершен ли обход.

Упражнение 4.1.2. Разобрать пример использования ArrayList<integer>. Пояснить, что в <> указывается тип, хранимый в списке. Данный тип обязан быть объектом. Разобрать две схемы прохода по контейнеру.

Последовательный и **двоичный** поиск. Последовательный поиск в произвольном линейном массиве: трудоёмкость n. Двоичный поиск в известной игре угадывания числа. Определение его трудоёмкости как двоичный логарифм n. Поиск в отсортированном массиве с помощью метода Arrays.binarySearch().

Упражнение 4.1.3. Практическое занятие по использованию методов класса Arrays, реализующих поиск.

Задание 4.1.1. Разработка собственного метода, например, вывод массива на печать. Продемонстрировать объектно-ориентированный подход к решению задаче, если тип элементов массива произвольный: необходимость наличия «печатающего» метода для класса, к которому принадлежат элементы массива.

Задание 4.1.2. Задана матрица, в которой элементы упорядочены при просмотре слева направо сверху вниз. Реализовать класс, который находит в такой матрице строку и столбец, в котором находится заданный элемент. Необходимо обработать ситуацию, когда элемент отсутствует.

Тема 4.2. Списки

Цель. Освоение понятия «список» как структуры данных, изучение библиотечного класса LinkedList.

Понятие Список. Разновидности структур данных, основанных на списках. Список как базовая структура данных. Классификация структур данных, основанных на списках: по дисциплине обслуживания (стеки, очереди), по структуре (односвязные и двусвязные списки).

Очередь как реализация принципа FIFO (первым пришёл – первым вышел). Работа с очередью осуществляется с обоих концов.

Стек как реализация принципа LIFO (последним пришёл – первым вышел). Работа со стеком осуществляется с одного конца. Обратить внимание на следующее различие: работать с очередью записывающий и считывающий процессы, как правило, могут асинхронно, т.е. один процесс пишет, другой видит, что что-то появилось, и забирает что бы то ни было. Считывание же вершины стека может происходить не в любой ситуации, а только если вершина стека находится в определённом состоянии. Если это состояние не достигнуто, продолжается запись в вершину стека.

Двусвязные и **односвязные** списки. Мотивация введения второго указателя: облегчение навигации по списку в обратном направлении, т.к. сдвиг в односвязном списке назад требует больших трудозатрат, в частности, перемещение назад от конца списка требует прохода от начала по всему списку.

Библиотечный класс LinkedList, Queue, Stack. Практическое занятие по библиотечному классу LinkedList, реализующему связные списки.

Класс LinkedList как реализация связного списка. Методы, специфичные для связного списка: addFirst, addLast,getFirst,getLast,removeFirst, removeLast.

Класс Stack и **интерфейс Queue**. Объяснение, что LinkedList – это одна из реализаций интерфейса очередь, привести примеры других реализаций.

Упражнение 4.2.1. Пример применения классов LinkedList, Queue и Stack.

Задание 4.2.1. Реализовать интерфейс очереди с ограничением на количество. Очередь должна игнорировать добавление элемента, если её размер достиг максимума. Требуется реализовать методы poll (извлечь первый элемент из очереди), peek (считать первый элемент из очереди) и add (добавить элемент в конец очереди).

Тема 4.3. Адаптеры в Android

Цель. Освоить применение адаптеров при разработке Android-приложений на Java.

Адаптеры. Для чего нужны адаптеры. Готовые адаптеры в Android: SimpleAdapter, ArrayAdapter. Абстрактный класс BaseAdapter.

Применение адаптеров для обработки событий пользовательского интерфейса. Отображение ListView через адаптер. Методы getView(), getListAdapter().

Упражнение 4.3.1. Разбор примера кода с реализацией ListView через ArrayAdapter.

Упражнение 4.3.2. С помощью SimpleAdapter реализовать более сложный список, в котором строка формируется из двух текстов.

Тема 4.4. Деревья.

Цель. Освоение понятия «дерево» как структуры данных, изучение библиотечного класса TreeSet.

Понятие дерево. Разновидности деревьев. Дерево, как базовая структура данных. Сбалансированные деревья. Двоичные деревья.

Понятие бинарного дерева. Акцентирование внимания на то, что дерево является самоподобной структурой, у которой часть подобна целому. Описание дерева на языке Си в виде структуры из трех полей: данные, указатель на левое поддерево, указатель на правое поддерево. Разбор рекурсивной программы, производящей обход дерева и вывод на печать его данных.

Понятие сбалансированного дерева. Полярный пример: дерево, вырожденное в список; дерево, у которого высота левого и правого поддерева одинаковы.

Библиотечный класс **TreeSet**. Порядок элементов в TreeSet, методы add, contains, floor, ceiling, subset.

Упражнение 4.4.1. Разобрать пример использования класса TreeSet.

Задание 4.4.1. Реализовать класс, который с использованием TreeSet находит список учеников, сдавших экзамен на оценку выше заданного числа. Для решения задачи нужно реализовать собственный класс Pupil, реализующий интерфейс Comparable.

Тема 4.5. Рекурсия

Цель. Приобрести представление о рекурсивных алгоритмах.

Понятие рекурсивного вызова в программировании. Умение видеть в задаче

ситуацию «часть подобна целому» и осмыслить тот факт, что единовременно работает множество экземпляров рекурсивной функции. Формальные и фактические параметры, локальные и глобальные переменные при перемещении по стеку рекурсивных вызовов.

Линейная и **ветвящаяся** рекурсия. Рассмотрение примеров функций с линейной и ветвящейся рекурсией. Иллюстрация «вручную» работы рекурсивной программы (для каждого экземпляра функции подписать передаваемые аргументы и возвращаемые результаты):

- с линейной рекурсией в виде цепочки;
- с ветвящейся рекурсией в виде дерева рекурсивных.

Демонстрация на полученных иллюстрациях необходимости нерекурсивной заглушки.

Упражнение 4.5.1. Пример линейной рекурсии. Вычисление степени n числа x. Описание рекурсивной функции. Формулировка окончания рекурсии. Подробная иллюстрация процесса вычисления. Демонстрация ограничений рекурсивных программ (ошибка переполнения стека).

Упражнение 4.5.2. Пример ветвящейся рекурсии – поиск файла в дереве директорий.

Задание 4.5.1. Для заданных программ определить вид рекурсии, нарисовать иллюстрацию рекурсивных вызовов, «вручную» определить количество рекурсивных вызовов.

Тема 4.6. Алгоритмы сортировки

Цель. Разобрать алгоритмы сортировки и подходы к их реализации.

Значение алгоритмов сортировки. Важность сортировки как самой по себе, так и для повышения быстродействия других алгоритмов: поиска, вставки, слияния.

Сортировка вставкой как наиболее простой алгоритм сортировки. Пошаговая реализация на примере. Программа, реализующая сортировку вставкой для массива целых чисел. Оценка трудоёмкости как n2.

Быстрая сортировка. Идея алгоритма быстрой сортировки (рекурсивное разбиение массива на два). Пошаговая реализация на примере. Структура программы, реализующей быструю сортировку. Оценка трудоёмкости как n*log(n).

Сортировка массивов с помощью метода Arrays.sort(). Уяснение того факта, что способ сравнения для класса, к которому относятся элементы массива, должен быть известен (для двух элементов нужно как-то определить, кто из них больше). Мотивация наследования интерфейса Comparable и реализации метода compareTo(), чтобы массив мог быть отсортирован.

Упражнение 4.6.1. Практическое занятие по использованию методов класса Arrays и Collections, реализующих сортировку.

Сортировка и **поиск** в List-контейнерах. Важно уяснить, что методы sort и binarySearch для списков являются статическими методами базового класса Collections, а не класса Arrays.

Задание 4.6.1. Реализовать функцию, которая проверяет является ли массив отсортированным.

Тема 4.7. Хэш-таблица и функция хэширования

Цель. Изучить поддержку хэширования в Java и назначение фундаментальной функции hashCode(), ознакомиться с разделением контейнерных классов в java на Collections и Map; освоить применение библиотечного контейнера Set.

Метод hashCode() как метод базового класса Object. Метод hashCode()

как реализация хэш-фукнции для контейнера, для которого необходимо обеспечить быструю выборку элементов. Необходимость наличия эффективной фукнции hashCode() для реализации класса, представляющего хэш-таблицу.

Понимание факта, что универсального метода для написания метода hashCode() не существует: алгоритм его вычисления должен основываться на семантике объектов класса, для которого этот метод необходимо перегрузить. Перечисление руководящих принципов написания hashCode():

- возвращение одного и того же значения для одного и того же объекта при каждом вызове;
- вычисление хэш-кода должно основываться исключительно на содержимом объекта, но не на его случайной уникальной информации (например, адресе в памяти);
- множество значений, генерируемое функцией hashCode() на множестве объектов контейнера, должно быть распределено как можно равномернее, чтобы каждому значению соответствовало примерно равное количество объектов контейнера.

Упражнение 4.7.1. Пример реализации метода hashCode() для строкового контейнера: хэш-код элемента контейнера равен стандартному хэш-коду данной строки, умноженному на количество её повторений в контейнере (текст примера приведён в приложении).

Семейства контейнеров – Collections и Map. Наличие в библиотеке двух семейств контейнеров – Collections и Map, и подразделение Collection, в свою очередь, на List и Set. Отличие простых контейнеров от ассоциативных. Общая иерархическая схема контейнерных библиотечных классов Java.

Контейнер Set. Set как контейнер уникальных объектов. Наличие двух реализаций: HashSet и TreeSet. Необходимость наличия метода equals() для обоих.

Контейнер HashSet. Уяснение принципиального требования: для класса, объекты которого помещаются в данный контейнер, должен быть написан метод hashCode().

Упражнение 4.7.2. Пример, демонстрирующий отличие между контейнерами: заполнение HashSet и TreeSet одним и тем же набором строковых элементов в одной и той же последовательности; обход содержимого с помощью итератора и вывод на печать; обнаружение разного порядка хранения элементов в этих контейнерах.

Задание 4.7.1. Реализовать собственный класс PairInteger. Сделать так, чтобы его можно было использовать в HashSet (реализовать методы hashCode и equals) и в TreeSet (реализовать метод compareTo).

Тема 4.8. Ассоциативные массивы

Цель. Освоить понятие «ассоциативный массив» и его реализацию в Java.

Ассоциативный массив как набор пар «ключ-значение». Требование уникальности ключа. Примеры, когда естественным индексом выступает не целое число, а произвольная символьная строка (например, «государство – столица»). Ознакомление учащихся с фактом, что многомерный ассоциативный массив с произвольным набором индексов, называемый глобал, выступает в качестве основной структуры данных в нереляционных БД, и для работы с ним существует набор специальных функций.

Общие методы Map: put(), get(), containsKey(), containsValue().

Контейнер HashMap. Контейнер HashMap и пример его использования. Идея контейнера: поиск ключей с помощью хэш-кодов значений.

Контейнер TreeMap. Контейнер TreeMap, хранение данных, упорядоченных по ключу. Метод subMap(), возвращающий часть дерева.

Потоко-безопасность объектов. Напоминание понятия потоко-безопасности объекта. Возможность сделать контейнер потоко-безопасным. Статические методы класса Collections: synchronizedCollection, synchronizedList, synchronizedSet, synchronizedMap.

Упражнение 4.8.1. Разобрать пример использования TreeMap и HashMap.

Задание 4.8.1. Реализовать многопоточную программу, которая добавляет в synchronizedMap элементы с ключами от 100 * THREAD_NUM до 100 * (THREAD_NUM + 1), где THREAD_NUM – номер потока. Изучить, что происходит, если использовать обычный TreeMap.

Тема 4.9. Реляционная модель данных. СУБД

Цель. Рассказать учащимся, что же представляет собой в общих чертах предмет изучения, зачем он нужен и какова история его развития. Понять представление реалий окружающего мира с помощью модели «сущность-связь».

Получить представление о реляционной модели данных и получить представление о записи связей в реляционной базе.

Важность БД и СУБД. Использование БД во всех направлениях современной деятельности человека на примерах из повседневной жизни. Необходимо объяснить, почему хранение данных в обычных текстовых файлах не может считаться приемлемым решением для эффективной работы с данными и какие задачи в этой связи возлагаются на СУБД. История развития и классификация СУБД.

Характеристика СУБД как соединения программного обеспечения и данных и неотъемлемо включающей следующие составные части:

- набор файлов, содержащих данные – непосредственно физическая база данных;
- спецификация информационного содержимого физической базы данных – схема данных;
- программное обеспечение, поддерживающее доступ и изменение содержимого базы данных – механизм СУБД;
- язык программирования СУБД, используемый для задания схемы и осуществления доступа к базе данных.

Хранение данных в таблицах. Привести примеры хранения данных в таблице. Описание типов связи и отношений: один к одному, один ко многим, многие ко многим. Примеры отношений: муж – жена (в России в каждый момент времени может быть только один супруг), сын – отец (отец всегда один, детей много), брат – сестра (каждый может иметь много братьев и сестер).

Объяснить способы хранения таких данных в таблицах. Способы хранения дополнительных данных в отношениях.

Мини-проект 4.1. Записная книжка. Спроектировать БД для приложения «Записная книжка» на бумаге.

Тема 4.10. Локальные СУБД. Введение в SQL

Цель. На примере локальной СУБД SQLite изучить средства SQL, используемые для создания и модификации содержимого таблиц, изменения их структуры, удаления.

Локальная СУБД SQLite. Знакомство с локальной СУБД SQLite.

Минипроект 4.1. Записная книжка. Создать БД SQLite «Записная книжка» по спроектированной ранее структуре. На её примере разобрать все изучаемые далее инструкции SQL, создание простейшего Android-приложения. Самостоятельно закончить

мини-проект.

Язык запросов SQL. О языке запросов SQL. Создание таблиц. Синтаксис запроса CREATE TABLE, используемого для создания таблицы. Команды ALTER TABLE и DROP TABLE, SHOW TABLES.

Для создания базы данных SQLite проще всего воспользоваться утилитой SQLiteStudio.

Вставка, **изменение** и **удаление** данных из таблицы. Краткое разъяснение и синтаксис команды (INSERT INTO table VALUES …).

Обновление таблиц. Синтаксис запроса UPDATE, используемого для изменения записей таблиц. Ключевое слово WHERE и простейшие фильтры.

Выборка данных. Краткое разъяснение назначения и синтаксис команды SELECT. Форма SELECT * для просмотра всей таблицы. Форма SELECT для просмотра только определённых столбцов таблицы. Ключевое слово FROM; имя таблицы, которая является источником информации для запроса. Команда DELETE FROM table WHERE.

Булевы операции в запросах. Более сложные запросы SELECT. Ключевые слова ORDER BY и DISTINCT.

Агрегация. Вариант SELECT для подсчёта количества, суммы, максимума и минимума, среднего и других агрегаций.

Практикум

Практическое занятие по темам модуля. Конкретное содержание определяется учителем.

Контрольное тестирование по модулю

Электронный тест охватывает все темы модуля и в большей части направлен на оценку практических знаний и навыков учеников, полученных в ходе изучения Модуля.

Модуль 5. Основы разработки серверной части мобильных приложений

Тема 5.1. 2 часа. IP-сети.
Тема 5.2. 4 часа. Web-сервер. HTTP-запросы и ответы.
Тема 5.3. 4 часа. Клиент-серверная архитектура мобильных приложений.
Тема 5.4. 4 часа. Облачные платформы. REST-взаимодействие.
Тема 5.5. 6 часов. Серверные СУБД.
Практикум. 6 часов.
Контрольное тестирование по модулю. 2 часа.

Тема 5.1. IP-сети

Цель. Ознакомиться с общим устройством сетей. Осознать понятие IP-адреса и способы его присвоения. Получить представление об общей схеме передачи данных по сети. Понять соответствие IP-адреса и адреса сайта. Освоить наиболее популярные сетевые команды и понаблюдать за их выполнением.

Общие понятия. Интернет и протоколы TCP/IP. Адресация устройств в сетях: для чего она нужна. Что может иметь собственный IP-адрес. Кем назначается IP-адрес и как он выглядит в IPv4. Понятие статического и динамического назначения (DHCP). Понятие внешнего и внутреннего IP-адреса. Знакомство с сервисом http://www.nic.ru/whois/ для определения информации о внешних IP-адресах. Исчерпание IPv4-адресов (о новых возможностях IPv6). Маски подсети. Доменные имена (DNS). URL-ссылки. IP-адрес.

Стек протоколов TCP/IP. Порты. Маршрутизация в TCP/IP. DNS. Whois-сервис. URL-ссылка. Уяснить разницу между понятиями сайт, сервер, доменное имя.

Упражнение 5.1.1. Узнать собственный внешний IP-адрес с помощью сервиса http://2ip.ru/. Пояснить, что внешний IP-адрес у всех в комнате одинаковый, так как это адрес компьютера, подключенного к интернету.

В сервисе http://www.nic.ru/whois/ изучить информацию о собственном внешнем IP-адресе. Можно также посмотреть информацию об IP-адресах: 8.8.8.8, 87.240.131.99, 211.45.27.198 или 173.194.71.113. Пояснить, что некоторые серверы могут иметь много IP-адресов, часть пользователей направляется на один сервер, а часть на другой.

Команда ping. Синтаксис. Сценарии поведения, когда связь с адресатом есть и когда её нет. Использование для определения IP-адреса домена и для оценки скорости соединения.

Команда tracert. (traceroute в Linux). Синтаксис. Наблюдения за трассами до различных хостов. Желательно до урока подобрать примеры таких адресов.

Команда nslookup. Синтаксис. Перевод ip-адреса из цифрового вида в доменный и обратно.

Команды ipconfig (ifconfig в Linux). Синтаксис. Команда позволяет узнать свой внутренний IP-адрес, адрес маршрутизатора. Уточнить, что у одного компьютера может быть несколько IP-адресов (для каждого сетевого устройства).

Упражнение 5.1.2. Все команды должны опробоваться школьниками сразу после описания их синтаксиса. В процессе выполнения записать IP-адреса популярных сервисов и время ответа на команду ping.

Задание 5.1.1. Опробовать аналогичные команды на домашнем компьютере. Сравнить записанные значения в компьютерном классе со значениями дома. Объяснить, почему значения существенно отличаются.

Тема 5.2. Web-сервер. HTTP-запросы и ответы

Цель. Получить общее представление о структуре HTTP-запроса и ответа сервера.

Основные понятия. Структура HTTP-запроса: метод, заголовок, тело. Продемонстрировать HTTP через команду telnet, чтобы показать, что HTTP – обычный текстовый протокол, который, как правило, использует порт 80. Понятие сокета.

Web-сервер. Понятие web-сервера, разбор на примере одной из наиболее популярных и распространенных программ, созданных для веб-разработчиков.

Методы GET и POST. Различие GET и POST запросов следуют из их назначения:

- POST – предназначен для передачи данных серверу и влечёт изменение данных на сервере (например, регистрация пользователя);
- GET – также передаёт данные серверу, но эти данные не влекут за собой значительных изменений на сервере (например, передача параметров поиска, активация аккаунта).

Как следствие, у запроса GET тело отсутствует, у запроса POST оно содержит данные.

Примеры запросов GET и POST, передающих на сервер одну и ту же информацию (необходимо подчеркнуть различие, как выглядит эта информация в запросе GET и POST).

Ответы сервера. Типовая структура ответа сервера. Классификация кодов ответов: информационные (от 100 до 200), запрос клиента успешен (от 200 до 300), запрос клиента переадресован (от 300 до 400), ошибка клиента (от 400 до 500), ошибка сервера (более 500). Объяснение наиболее распространённых ответов: 200, 301, 302, 401, 404, 408, 500,

502, 503, 505.

Упражнение 5.2.1. Посмотреть, как внутри устроены запросы и ответы к серверу. Открыть страницу ya.ru (или любую другую на выбор). Открыть в браузере «Developer Tools» (ctrl+shift+i в Google Chrome). Перейти во вкладку «Network». Возможно, стоит перезагрузить страницу, чтобы появилась информация. Пояснить, что каждая строчка – это некоторый запрос к серверу. В современном мире даже самая простая страница отправляет десяток запросов к серверу. Кликнуть на ya.ru для просмотра основного запроса, инициализированного вводом ссылки в браузер.

Посмотреть различные вкладки:

Headers – заголовки отправленного на сервер запроса (Request) и полученного от сервера ответа (Response). Можно нажать на кнопку «view source», чтобы посмотреть, как это выглядит на низком уровне.

Response – тело ответа сервера.

Timing – отображение времени, которое потребовалось на различные действия. Из интересного можно обратить внимание на DNS lookup (время поиска IP-адреса по доменному имени).

Отправка запроса на сервер с помощью формы из браузера. Описание формы в html. Тэги form, input (с различными типами). Значения и установка атрибутов action и method.

Упражнение 5.2.2. Задание:

1. Создание простой формы, которая отправляет значения двух полей «Имя» и «Фамилия» на сервер по кнопке «Отправить».

2. Реализация веб-сервера, который возвращает строку в наш браузер в формате согласно примеру. На странице заполнены поля:

Фамилия: Иванов

Имя: Пётр

Сервер возвращает строку «Иванов П.» и она отображается на созданной веб-странице.

Тема 5.3. Клиент-серверная архитектура мобильных приложений

Цель. Получить практический навык разработки простейших клиент-серверных мобильных приложений с использованием формата JSON для передачи сообщений.

Общие понятия. Структура и схема взаимодействия в клиент-серверных мобильных приложениях. Стандартные способы реализации. Синхронные и асинхронные запросы.

Отправка запросов из Android-приложений. Научиться отправлять запрос из Android-приложения. Для получения доступа в интернет необходимо добавить в манифест <*uses-permission android:name = "android.permission.INTERNET"/*>

Классы HttpClient, HttpResponse, StatusLine.

Упражнение 5.3.1. Разобрать программу для Android, которая реализует задание, аналогичное **Упражнению** 5.2.2. Использовать AsyncTask, чтобы не блокировать пользователя.

Формат JSON и **XML**. Сериализация. Формат JSON, синтаксис, применение. Сравнение с XML. Понятие сериализации. Сохранение сериализованного объекта в файл.

Библиотека Retrofit. Показать преимущества библиотеки Retrofit:

- отсутствие необходимости в реализации запросов к API в отдельном потоке;
- простота кода;
- подключение стандартных библиотек (например, Gson) для автоматической

конвертации JSON в объекты;

- динамическое построение запросов;
- обработка ошибок.

Упражнение 5.3.2. Модифицировать программу из упражнения 5.3.1 с использованием JSON и библиотеки Retrofit.

Тема 5.4. Облачные платформы. REST-взаимодействие

Цель. На практике освоить использование облачной платформы и REST-взаимодействия с сторонними сервисами.

Облачные платформы. Назначение и возможности облачных платформ: предоставление провайдером хостинга с готовым набором операционных систем, систем управления базами данных, связующему программному обеспечению, средств разработки и тестирования.

Бесплатные облачные платформы как доступный способ разработки готового к применению клиент-серверного приложения. Обзор возможностей на примере https://www.heroku.com/: языки программирования, СУБД, библиотеки.

Стиль взаимодействия REST. Общие принципы организации взаимодействия приложения с сервером посредством протокола HTTP в стиле REST. Важный принцип REST – сервер не запоминает состояние пользователя между запросами, каждый раз необходимо передавать все необходимые параметры. Понятие token.

Основные методы HTTP-запроса, которые реализуют операции: получение – GET, добавление – POST, модификация – PUT, удаление – DELETE.

Упражнение 5.4.1. Использование на разборе примера одного из API Яндекс https://tech.yandex.ru/#catalog, например, Предиктор.

OAuth-авторизация через REST API социальных сетей. OAuth-протокол. Объяснение назначения, принципа работы на примере «ВКонтакте». Token – текстовый ключ, который предоставляет ограниченный временный доступ к данным пользователя в социальной сети.

Упражнение 5.4.2. Задание: создать клиент-серверное приложение, которое позволяет авторизоваться через OAuth-авторизацию социальной сети «ВКонтакте» и отправлять информацию о пользователе на сервер.

Тема 5.5. Серверные СУБД

Цель. Получение навыков реализации Android-приложений с использованием серверных СУБД и OAuth-авторизации.

Серверные СУБД. Общий обзор серверных СУБД. Возможности. Архитектура клиент-серверных приложений, включающих СУБД. Необходимость промежуточного звена между СУБД и приложением-клиентом.

Практикум. Задание: создать клиент-серверное приложение, которое позволяет:

- авторизоваться через OAuth-авторизацию социальной сети «ВКонтакте»;
- сохранять информацию о пользователе и его местоположении в БД на сервере;
- отображать информацию о других пользователях;
- отображать местоположение всех пользователей на карте, при этом ограничить показ их числа заданным количеством.

Использование SQLite в клиент-серверных приложениях. Необходимость использования локальной СУБД для надёжного сохранения данных приложения

в случае неустойчивого канала интернет. Синхронизация локальной БД с серверной.

Индексы. Индексы необходимы для ускорения работы базы данных. Функционально работа базы данных не изменится, но скорость при больших объёмах данных ускорится во много раз. Команда CREATE INDEX.

Практикум

Практическое занятие по темам программы. Конкретное содержание определяется учителем.

Контрольное тестирование по модулю

Электронный тест охватывает все темы модуля и в большей части направлен на оценку практических знаний и навыков учеников, полученных в ходе изучения модуля.

3. Планируемые результаты

Предметные результаты:

- знание и соблюдение требований техники безопасности и санитарно-гигиенических норм;
- знание основ языка программирования Java и языка разметки XML;
- понимание принципа работы баз данных и клиент-серверных протоколов;
- умение использовать разные алгоритмы в приёмах программирования,
- умение пользоваться ПК и IDE-разработки для программирования устройства;
- умение читать готовую программу и находить ошибки в готовых программах.

Личностные результаты:

- формирование ответственного отношения к учению, готовности и способности обучающихся к саморазвитию и самообразованию, средствами информационных технологий;
- формирование универсальных способов мыслительной деятельности (абстрактно-логического мышления, памяти, внимания, творческого воображения, умения производить логические операции);
- развитие опыта участия в социально значимых проектах, повышение уровня самооценки благодаря реализованным проектам;
- формирование коммуникативной компетентности в общении и сотрудничестве со сверстниками в процессе образовательной, учебно-исследовательской и проектной деятельности;
- формирование целостного мировоззрения, соответствующего современному уровню развития информационных технологий;
- формирование осознанного позитивного отношения к другому человеку, его мнению, результату его деятельности;
- формирование ценности здорового и безопасного образа жизни; усвоение правил индивидуального и коллективного безопасного поведения при работе с компьютерной техникой.

Метапредметные результаты:

- ориентироваться в своей системе знаний: отличать новое знание от известного;
- умение производить анализ поставленной задачи, самостоятельно решать её; формулировать, аргументировать и отстаивать свое мнение; извлекать нужную информацию из открытых источников; составлять примерный алгоритм работы.

ОРГАНИЗАЦИОННО-ПЕДАГОГИЧЕСКИЕ УСЛОВИЯ РЕАЛИЗАЦИИ ОБЩЕРАЗВИВАЮЩЕЙ ПРОГРАММЫ

1. Условия реализации общеразвивающей программы

Материально-техническое обеспечение

Требования к помещению:

- компьютерный класс, отвечающий требованиям СанПиН для учреждений дополнительного образования;
- кабинет с 14 рабочими местами для обучающихся, рабочим местом преподавателя;
- качественное освещение.

Оборудование:

- настольный ПК с программным обеспечением Eclips, Android Studio, объединенные в локальную сеть;
- планшет (для отладки);
- интерактивная доска;
- ПК для педагога, объединённый с функцией сервера.

Раздаточный материал:

- учебный материал по теме (портал обучения myitschool.ru);
- демонстрационные программы;
- инструкции по настройке среды разработки.

2. Формы аттестации

Аттестация проводится в форме тестирования после освоения каждого модуля. Оценка производится на основе критериального оценивания.

Итоговая аттестация учащихся осуществляется по 100-балльной шкале, которая переводится в один из уровней освоения образовательной программы согласно таблице:

Набранные баллы учащимся	**Уровень освоения**
0–50 баллов	низкий
50–75 баллов	средний
75–100 баллов	высокий

3. Фонд оценочных материалов

Распределение баллов и критерии оценивания

№ п/п	**Название модуля**	**Количество баллов**	
		минимальное	**максимальное**
1.	**Модуль 1. Основы программирования**	**5**	**10**
	Посещение занятий	2	3
	Проектная деятельность	3	7
2.	**Модуль 2. Объектно-ориентированное программирование**	**7**	**15**
	Посещение занятий	2	5
	Проектная деятельность	5	10

3.	**Модуль 3. Основы программирования Android-приложений**	**7**	**15**
	Посещение занятий	2	5
	Проектная деятельность	5	10
4.	**Модуль 4. Алгоритмы и структуры данных**	**8**	**20**
	Посещение занятий	2	8
	Проектная деятельность	6	12
5.	**Модуль 5. Основы разработки серверной части мобильных приложений**	**8**	**20**
	Посещение занятий	2	8
	Проектная деятельность	6	12
6.	**Итоговая защита**	**10**	**20**
	Посещение занятий	4	8
	Проектная деятельность	6	12
Итого:		**45**	**100**

4. Методические материалы

Образовательный процесс осуществляется в очной форме.

В образовательном процессе используются следующие **методы обучения**:

- устные (беседы, объяснение);
- поисковые (изменение программы для приобретения устройством новых свойств);
- демонстрационные (демонстрация возможностей устройства);
- практические (написание программы, проведение мини-соревнований).

Программа обучения состоит из нескольких основных блоков:

- обучение основам языка Java, работе с БД и клиент-серверной архитектурой; базовым принципам ООП;
- знакомство с Android.

Программой предусмотрены следующие виды деятельности обучающихся:

- работа с технической и справочной литературой;
- программирование;
- эксперимент, испытание.

Методическое обеспечение

Методические пособия, разрабатываемые преподавателем с учётом конкретных условий. Техническая библиотека объединения, содержащая справочный материал, учебную и техническую литературу. Индивидуальные задания.

Методическое обеспечение учебного процесса включает разработку преподавателем методических пособий, вариантов демонстрационных программ и справочного материала:

- демонстрационные программы;
- инструкции по настройке среды разработки;
- справочные материалы по терминам ПО.

Список литературы

1. Блох Джошуа. Java. Эффективное программирование. Effective Java. Programming Language Guide. изд. «Лори». 2014 г. 310 стр. ISBN 978-5-85582-347-9.
2. Гослинг Джеймс, Билл Джой, Гай Л. Стил, Гилад Брача, Алекс Бакли. Язык программирования Java SE 8. Подробное описание. The Java Language Specification: Java SE8 Edition.изд. «Вильямс». 2015 г. 672 стр. ISBN 978-5-8459-1875-8, 978-0-13-390069-9.
3. Зигард Медникс, Лайрд Дорнин, Блейк Мик, Масуми Накамура. Программирование под Android. Programming Android. изд. Питер. 2012 г. 496 стр. ISBN 978-5-459-01115-9, 978-1-449-38969-7.
4. Майер Рето. Android 2. Программирование приложений для планшетных компьютеров и смартфонов. Professional Android 2: Application Developmentecond Edition. изд. Эксмо. 2011 г. 672 стр. ISBN 978-5-699-50323-0.
5. Харди Брайн, Билл Филлипс.Программирование под Android. Android Programming: The Big Nerd Ranch Guide. изд. Питер. 2014 г. 592 стр. ISBN 978-5-496-00502-9, 978-0-321-80433-4.
6. Санитарно-эпидемиологические правила и нормативы СанПиН 2.4.4.3172-14.
7. Распоряжение Правительства РФ от 04.09.2014 № 1726-р «Об утверждении Концепции развития дополнительного образования детей».
8. Портал обучения https://myitschool.ru.
9. Федеральный закон от 29.12.2012 N 273-ФЗ (ред. от 31.12.2014, с изм. от 02.05.2015) «Об образовании в Российской Федерации» (с изм. и доп., вступ. в силу с 31.03.2015).

Аннотация

Программа «Мобильная разработка» была составлена при поддержке © Samsung R&D Institute Rus (Исследовательский Центр Samsung) и при участии Московского физико-технического института (МФТИ).

Проект «IT-школа Samsung» компании Samsung Electronics – это долгосрочная инициатива, которая реализуется при поддержке Министерства образования и науки РФ. В её рамках запланировано бесплатное обучение более 5 тысяч школьников в 20 регионах России по программе дополнительного образования технической направленности в области IT и программирования на языке Java.

Данная программа является единственным в своём роде экспериментом в связи с востребованностью на рынке и отсутствием программ образования в данном направлении для школьников. Особенность программы «Мобильная разработка» заключается в изучении основ языка программирования Java и структуры приложения под ОС Android и строится в доступной и понятной для учащихся среде, т.е. программирование ведётся в текстово-графическом режиме, что позволяет сразу задавать необходимый функционал для элементной базы приложения.