Муниципальное общеобразовательное учреждение
«Средняя общеобразовательная школа №3 г. Ершова Саратовской области»

ПРИНЯТА	УТВЕРЖЕНО
на заседании педагогического совета МОУ «Средняя общеобразовательная Школа №3 г. Ершова Саратовской области» Протокол № 13 от 19.04.2023	директор МОУ «Средняя общеобразовательная Школа №3 г. Ершова Саратовской области» Приказ № 140 от 24.04.2023 _____ А.В. Широкова

Дополнительная общеобразовательная общеразвивающая программа технической направленности «Мобильная разработка» (стартовый уровень)

Направленность: техническая
Форма реализации: очная
Возраст обучающихся: 8 - 11 лет
Срок реализации: 1 год

МУНИЦИПАЛЬНОЕ ОБЩЕОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ " СРЕДНЯЯ ОБЩЕОБРАЗОВАТЕЛЬНАЯ ШКОЛА № 3 Г. ЕРШОВА САРАТОВСКОЙ ОБЛАСТИ"

Подписан: МУНИЦИПАЛЬНОЕ ОБЩЕОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ " СРЕДНЯЯ ОБЩЕОБРАЗОВАТЕЛЬНАЯ ШКОЛА № 3 Г. ЕРШОВА САРАТОВСКОЙ ОБЛАСТИ"
Основание: Я являюсь автором этого документа
Местоположение: место подписания
Дата: 2023.04.24 16:59:22+04'00'

Автор – составитель:
Бурова Ольга Валерьевна – педагог дополнительного образования

Ершов, 2023

Раздел №1 «Комплекс основных характеристик программы»

Пояснительная записка

Дополнительная общеобразовательная общеразвивающая программа «Мобильная разработка» (стартовый уровень) муниципального общеобразовательного учреждения «Средняя общеобразовательная школа № 3 г. Ершова Саратовской области» разработана в рамках **технической направленности в** соответствии со следующими документами:

- «Закон об образовании в Российской Федерации» (№ 273-ФЗ от 29 декабря 2012 г.);
- «Порядок организации и осуществления образовательной деятельности по дополнительным общеобразовательным программам» (пр. Министерства просвещения РФ от 27 июля 2022 г. №629)
- Приказ Министерства образования и науки Российской Федерации от 23 августа 2017 года №816 «Об утверждении Порядка применения организациями, осуществляющими образовательную деятельность, электронного обучения, дистанционных образовательных технологий при реализации образовательных программ»;
- Постановление Главного государственного санитарного врача РФ от 28 сентября 2020 года №28 «Об утверждении СанПиН 2.4.4.3648-20 «Санитарно-эпидемиологические требования к организациям воспитания и обучения, отдыха и оздоровления детей и молодёжи»
- Положением о дополнительной общеобразовательной общеразвивающей программе МОУ «СОШ № 3 г. Ершова Саратовской области»

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 своихприложений. 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.

Дополнительная общеобразовательная общеразвивающая программа «Мобильная разработка» ориентирована на развитие навыков программирования и проектирования программ под платформу Android.

Актуальность программы образовательной программы связана с тем, что в настоящее время широкое распространение получили мобильные устройства: планшеты, смартфоны, и др. Количество мобильных устройств значительно превысило количество настольных компьютеров и ноутбуков,

их возможности уже приближаются к возможностям современных компьютеров по быстродействию и объему памяти. Значительное число новых информационныхсистем и программных продуктов разрабатывается с учетом возможности работы на мобильных устройствах.

Новизна. Программа «Мобильная разработка» отражает требования и актуальные тенденции не только сегодняшнего, но и завтрашнего дня, а также имеет междисциплинарный характер, что полностью отражает современные тенденции построения как дополнительных общеобразовательных программ,так и образования в целом.

Также данная программа является базой для перехода на более сложные программы обучения. Обучающиеся приобретают знания по основам IT, которые будут востребованы для дальнейшего обучения в профильных среднихспециальных и высших учебных заведениях.

Отличительные особенности. В основу данной программы положена программа Шанина М.М. и Петраковой Т.В. «Мобильная разработка (стартовый, базовый уровень)» г Екатеринбург, 2021г. В отличие от базовой изменен объем программы (со 114 часов на 144), возраст обучающихся.

Программа предполагает использование и реализацию общедоступных и универсальных форм организации материала, минимальную сложность предлагаемого материала для освоения содержания программы в начале обучения курса и 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 в конце обучения.

Педагогическая целесообразность. Программа реализует профориентационные задачи, обеспечивает знакомство с современными профессиями в сфере IT. Осваивая данную программу, обучающиеся будут овладевать навыками, востребованными на рынке труда практически для каждой перспективной профессии.

Адресат программы. Дополнительная общеразвивающая программа «Мобильная разработка» предназначена для детей в возрасте 10–18 лет, без ограничений возможностей здоровья. Количество обучающихся в группе – до 12 человек. Состав групп постоянный. Программа ориентирована на учащихся в возрасте 10–18 лет, которые:

- имеют склонность к алгоритмическому мышлению;
- увлекаются ИТ-технологиями;
- владеют хотя бы одним языком программирования;
- имеют устойчивые знания по школьному курсу математики за 1–8 класс;
- уверенно владеют двоичной системой счисления, переводом чисел между десятичной, двоичной, восьмеричной и шестнадцатеричной системами счисления, сложением и вычитанием в них;
- знают основы логики, теории множеств и операций над ними.

Возрастные особенности. Дети этого возраста отличаются внутренней уравновешенностью, стремлением к активной практической деятельности, поэтому основной формой проведения занятий выбраны практические

занятия. Ребят также увлекает совместная, коллективная деятельность, так как резко возрастает значение коллектива, общественного мнения, отношений со сверстниками, оценки поступков и действий ребёнка со стороны не только старших, но и сверстников. Ребёнок стремится завоевать в их глазах авторитет, занять достойное место в коллективе. Поэтому в программу включены практические занятия соревновательного характера, которые позволяют каждому проявить себя и найти своё место в детском коллективе.

Также следует отметить, что дети данной возрастной группы характеризуются такими психическими процессами, как изменение структуры личности и возникновение интереса к ней, развитие абстрактных форм мышления, становление более осознанного и целенаправленного характера деятельности, проявление стремления к самостоятельности и независимости, формирование самооценки. Эти процессы позволяют положить начало формированию начального профессионального самоопределения обучающихся.

Наполняемость группы: 12-15 человек

Срок реализации и объем программы: 1 год, 144 часа.

Режим занятий. Занятия проводятся 2 раза в неделю по 2 часа. Продолжительность занятий 45 минут с перерывом длительностью не менее 10 минут.

Так как в течение учебного года возникает непреодолимая сила, или форс-мажор – обстоятельства (эпидемия, карантин, погодные условия и прочее), не позволяющие осуществлять обучение в обычной (очной) форме, реализация программы возможна с помощью электронных (дистанционных) технологий.

1.2. Цель и задачи программы

Цель*:* формирование технической грамотности средствами приобщения обучающихся к разработке программ под современную платформу Android.

Задачи:

Обучающие:

- расширить знания о современных и популярных платформах;
- обучить языку программирования Java, языку разметки XML;
- обучить объектно-ориентированному подходу в проектировании и разработке программного обеспечения;
- познакомить с архитектурой приложения под Android;
- обучить программированию технических устройств.

Развивающие:

- сформировать алгоритмическое мышление;
- развить логическое и техническое мышление;
- сформировать навыки работы с информацией;
- сформировать умение самостоятельно решать поставленную задачу;
- сформировать умение излагать мысли в чёткой логической последовательности, отстаивать свою точку зрения, анализировать ситуацию и самостоятельно находить ответы на вопросы путём логических рассуждений;

– сформировать умение планировать свои действия с учётом фактора времени, в обстановке с элементами конкуренции, предвидеть результат и достигать его, при необходимости вносить коррективы в первоначальный замысел.

Воспитательные:

– воспитать этику групповой работы, отношений делового сотрудничества,взаимоуважения;
– развивать основы коммуникативных отношений внутри проектных группи в коллективе в целом;
– воспитать упорство в достижении результата;
– сформировать целеустремлённость, организованность, ответственное отношение к труду, толерантность, уважительное отношение к окружающим.

1.3. Планируемые результаты

Личностные результаты

– сформированное ответственное отношение к учению, готовность и способность обучающихся к саморазвитию и самообразованию, средствами информационных технологий;
– сформированы универсальные способы мыслительной деятельности (абстрактно-логического мышления, памяти, внимания, творческого воображения, умения производить логические операции);
– развит опыт участия в социально значимых проектах, повышение уровня самооценки благодаря реализованным проектам;
– сформированы коммуникативные компетентности в общении и сотрудничестве со сверстниками в процессе образовательной, учебно-исследовательской и проектной деятельности;
– сформировано целостное мировоззрение, соответствующее современному уровню развития информационных технологий;
– сформировано осознанное позитивное отношение к другому человеку, его мнению, результату его деятельности
– усвоены правила индивидуального и коллективного безопасного поведения при работе с компьютерной техникой

Метапредметные результаты

– умение самостоятельно ставить и формулировать для себя новые задачи, развивать мотивы своей познавательной деятельности;
– умение самостоятельно планировать пути решения поставленной проблемы для получения эффективного результата;
– умение критически оценивать правильность решения учебно-исследовательской задачи;
– умение формулировать, аргументировать и отстаивать своё мнение;
– владение основами самоконтроля, способность к принятию решений;
– умение извлекать нужную информацию из открытых источников;
– умение организовывать учебное сотрудничество и совместную деятельность с педагогом и сверстниками в процессе проектной и

учебно- исследовательской деятельности.

Предметные результаты

- знание и соблюдение требований техники безопасности;
- 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.

1.4. Содержание программы

Учебный план дополнительной общеобразовательной общеразвивающей программы технической направленности «Мобильная разработка» (стартовый уровень)

№	Наименование раздела, темы	Количество часов			Формы аттестации/ контроля
		Теория	Практика	Всего	
1	***Основы программирования***	***11***	***21***	***32***	
1.1	Вводное занятие. Инструктаж по ТБ. Среда разработки	2	2	4	Знакомство. Опрос. Инструктаж по ТБ
1.2	Арифметика. Примитивные типы данных. Логика. Операции отношения и логические операции	3	5	8	Опрос
1.3	Условные конструкции. Блоки	1	3	4	Опрос
1.4	Итеративные конструкции. Массивы. Списки.	3	5	8	Опрос. Практическая работа
1.5	Методы (функции). Видимость переменных. Рекурсия.	2	4	6	Опрос
1.6	Практикум. Контрольное тестирование	-	2	2	Тестирование
2	***Объектно-ориентированное программирование***	***12***	***26***	***38***	
2.1	Классы и объекты	2	4	6	Опрос
2.2	Классы: конструкторы, статические методы. Начальные приёмы тестирования и отладки	2	4	6	Опрос

2.3	Android. Структура. Активности. Интерфейс пользователя. Язык разметки XML.ООП.	6	10	16	Опрос. Практическая работа
2.4	Намерения. Фрагменты.	2	6	8	Опрос. Практическая работа
2.5	Практикум. Контрольное тестирование.	-	2	2	Тестирование
3	***Основы программирования Android-приложений***	***8***	***26***	***34***	
3.1	Ввод, вывод и исключения	2	4	6	Опрос
3.2	Внутренние классы в обработке событий	2	4	6	Опрос
3.3	Параллелизм и синхронизация. Потоки	2	4	6	Опрос, практическая
					работа
3.4	Двумерная графика в Android-приложениях	1	5	6	Опрос
3.5	Реализация графики на основе SurfaceView	1	7	8	Опрос, практическая работа
3.6	Практикум. Контрольное тестирование	-	2	2	Тестирование
4	***Алгоритмы и структуры данных***	***6***	***24***	***30***	
4.1	Массивы. Списки. Алгоритмы сортировки. Алгоритм двоичного поиска. Деревья	2	6	8	Опрос, практическая работа
4.2	Адаптеры в Android	1	5	6	Практическая работа
4.3	Ассоциативные массивы	1	5	6	Практическая работа
4.4	Реляционная модель данных. СУБД. Введение в SQL	2	6	8	Опрос, практическая работа
4.5	Практикум. Контрольное тестирование	-	2	2	Тестирование
5	***Индивидуальное проектирование***	***4***	***6***	***10***	Защита проекта
	Итого:	**41**	**103**	**144**	

Содержание учебного плана дополнительной общеобразовательной общеразвивающей программы технической направленности «Мобильная разработка» (стартовый уровень)

Раздел 1. Основы программирования

Вводное занятие. Инструктаж по ТБ. Среда разработки

Теория. 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. 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.

Практика. 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.

Упражнение 2.1.2. Написать и отладить программу, которая выводила бы информацию об ученике в следующем виде: Ф.И.О., школа, класс.

Примитивные типы данных. Арифметика. Операции отношения и логические операции.

Теория. 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.

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

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, демонстрирующихприменение приоритетов.

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

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.

Практика. Упражнение 2.2.1. 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 запределы 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.

Упражнение 2.2.2. 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.

Упражнение 2.2.3. Написание простейших программ, демонстрирующих выполнение каждого изученного оператора. Объяснение результатов её выполнения.

Задание 2.2.1. 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 2.2.2. 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 2.2.3. 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++.

Упражнение 2.2.4. Программа, демонстрирующая выполнение логическихопераций и операций отношений. Объяснение результатов её выполнения.

Задание 2.2.4. Задачи на «ручное» написание логических выражений средствамиязыка 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.

Условные конструкции. Блоки

Теория. 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 вложенныхблоках.

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 конструкцийif-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 конструкцииswitch-case.

Практика.

Упражнение 2.3.1. Небольшие фрагменты кода, иллюстрирующие использование операторов ветвления, приоритетов вычисления операторов в выражении. Ускоренное вычисление логических выражений – прекращение вычислений, когда результат уже ясен.

Задание 2.3.1. Написать собственный пример на использование операторов ветвления. Например, нахождение максимума, минимума среди нескольких введённых переменных.

Итеративные конструкции. Массивы. Списки.

Теория. Внутренняя логика работы итеративных конструкций; их использование в различных формах, предусмотренных синтаксисом языка. Оператор for, for each, одномерные массивы. Понятие «список» как структура данных, библиотечный класс LinkedList.

Цикл с предусловием while. Синтаксис. Объяснение логики работы, примериспользования.

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

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

Массивы. Определение массива как совокупности элементов одного и того жетипа, расположенных вплотную друг за другом в памяти. Объявление массива двумяспособами. 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 eachперебирает не индексы массива, а сами элементы массива.

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 концасписка 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 длясвязного 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.

Практика. Упражнение 2.4.1. Небольшие фрагменты кода, иллюстрирующие использование операторов цикла (без использования массивов). Например, вычисление НОД по алгоритму Евклида.

Задание 2.4.1. Написать собственный пример на использование операторов цикла и операторов безусловного перехода. Например, проверка числа на то, что оно является простым.

Упражнение 2.4.2. Фрагменты кода, иллюстрирующие на одномерном массиве решение задач нахождения максимального, минимального.

Задание 2.4.2. 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 формуле.

Упражнение 2.4.3. Пример применения классов LinkedList, Queue и Stack.

Задание 2.4.3. 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).

Методы (функции). Видимость переменных. Рекурсия.

Теория. Определение функции как логически самостоятельной именованной части программы, которой могут передаваться параметры, и котораяможет 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.

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

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.

Практика. Упражнение 2.6.1. 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.

Упражнение 2.5.2. 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 2.5.3. Пример ветвящейся рекурсии – поиск файла в дереве директорий.

Задание 2.5.1. Для заданных программ определить вид рекурсии, нарисовать иллюстрацию рекурсивных вызовов, «вручную» определить количество рекурсивных вызовов.

Практикум. Контрольное тестирование

Практика. Практическое занятие по темам раздела. Тестирование по всем темам раздела направленное на оценку практических знаний и навыков.

Раздел 2. Объектно-ориентированное программирование

Классы и объекты.

Теория. 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.

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. Инкапсуляция: 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).

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

Практика. Упражнение 3.1.1. 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 средыразработки.

Упражнение 3.1.2. Разбор примера с применением классов Integer, Character,Float и т. д.

Упражнение 3.1.3. 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 3.1.1. Придумать примеры классов и соответствующие им примерыобъектов, полей и методов.

Классы: конструкторы, деструкторы и статические методы. Начальные приемы тестирования и отладки.

Теория. 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 классапри 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 кклассу (а нечлену класса) и его смысл.

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.

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 просмотрэтих 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 спомощью библиотеки JUnit.

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

Практика. Упражнение 3.2.1. 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.

Упражнение 3.2.2. 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 3.2.1. 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.

Упражнение 3.2.3. Разобрать примеры программ, продемонстрировать изученные приёмы тестирования.

Задание 3.2.2. Написать программу по обработке строк. При отладке использовать отладчик и изученные приёмы тестирования.

Архитектура приложений Андроид. Активности. Интерфейс пользователя. Язык разметки XML.

Теория. Знакомство со средой разработки приложений под Android. О среде разработки Android Studio. Порядок создания, компиляции, сборки и запуска в

среде. Порядок установки 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.

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.

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.

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

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.

Практика. Упражнение 3.3.1. Разбор кода простейшего Android-приложения, иллюстрирующего общую схему, его запуск. В материалах приведён пример приложения, решающего линейное уравнение.

Задание 3.3.1. Модифицировать приложение так, чтобы оно решало квадратное уравнение, а также корректно обрабатывало ситуацию нулевых коэффициентов.

Упражнение 3.3.2. Разобрать пример задания информации в xml (например, описание рецепта или геометрической фигуры).

Упражнение 3.3.3. Расположить кнопки в форме буквы П с помощью LinearLayout.

Упражнение 3.3.4. Разобрать работу функции println в классе Program в

testbed. Разобрать первые две строки process Button для получения входных и выходных данных в testbed.

Задание 3.3.2. 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 3.3.3. 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.

Упражнение 3.3.5. Разбор примера проектирования класса Дробь. Демонстрация преимуществ ОО-подхода.

Упражнение 3.3.6. 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.

Намерения

Теория. 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.

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: уэтого 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).

Практика. Упражнение 3.6.1. Разобрать пример с описанием классов, наследованием, переопределением метода, доступами и т. д.

Упражнение 3.4.2. Разобрать аналогичный пример, где одно из наследований заменено вложенностью классов. Объяснить различие в реализации. Задание 3.4.1. 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 compare To 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 методэкземпляра.

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 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).

Практикум. Контрольное тестирование

Практика. Практическое занятие по темам раздела. Тестирование по всем темам раздела направленное на оценку практических знаний и навыков.

Раздел 3. Основы программирования Android приложений

Ввод, вывод и исключения

Теория. 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.

Практика. Упражнение 4.1.1. Разбор примера кода с типовым использованием потоков ввода/вывода: чтение из файла и из памяти; вывод в файл. Использовать классы Scanner и PrintWriter.

Минипроект 4.1. Продолжение. Реализация обработки исключений в заготовке согласно заданию.

Задание 4.1.1. 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 количестворазличий.

Внутренние классы в обработке событий

Теория. 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.

Практика. Упражнение 4.2.1. Разбор примера кода с обработчиками событий.

Минипроект 4.1. Завершение. Реализация обработчиков событий с использованием анонимных классов согласно заданию.

Параллелизм и синхронизация. Потоки

Теория. 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 илиного 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 (). Методывзаимодействия потоков: suspend () – resume (), wait () -notify (). Метод join () каккоманда одному потоку дождаться завершения выполнения другого.

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

Практика. Упражнение 4.3.1. Разобрать пример программы, совершающей загрузку картинки из интернета и устанавливающей её на экран.

Двумерная графика в Android приложениях

Теория. Класс Canvas. Обзор методов и полей класса. Создание собственного View с методом onDraw. Вызов set ContentView 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.

Практика. Упражнение 4.4.1. 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 4.4.1. 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).

Разработка игровых приложений. Реализация графики на основе SurfaceView

Теория. Общие подходы для реализации игровых приложений. Последовательные этапы проектирования и реализации игрового приложения. Профессии в мире индустрии игр. Понятие игрового движка и его использованиепри 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. МетодыgetHolder(), lockCanvas(), unlockCanvasAndPost().

Практика. Упражнение 4.5.1. Разбор примера простейшей игры с анимацией.

Практикум. Контрольное тестирование

Практика. Практическое занятие по темам раздела. Тестирование по всем темам раздела направленное на оценку практических знаний и навыков.

Раздел 4. Алгоритмы и структуры данных

Массивы. Алгоритм двоичного поиска. Алгоритмы сортировки.

Теория. 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, чтопрограмма должна знать, как же ей сравнивать отдельные элементы массивов наравенство, 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.

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 = newArrayList<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. незавершен ли обход

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().

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

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 методаcompareTo(),чтобы массив мог быть отсортирован.

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.

Практика. Упражнение 5.1.1. Практическое занятие по библиотечному классу Arrays, реализующему массивы: заполнение, копирование, сравнение, печать, другие общие методы.

Упражнение 5.1.2. Разобрать пример использования ArrayList<integer>. Пояснить, что в <> указывается тип, хранимый в списке. Данный тип обязан быть объектом. Разобрать две схемы прохода по контейнеру.

Упражнение 5.1.3. Практическое занятие по использованию методов класса Arrays, реализующих поиск.

Задание 5.1.1. 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 5.1.2. 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.

Упражнение 5.1.4. Практическое занятие по использованию методов класса Arrays и Collections, реализующих сортировку.

Задание 5.1.3. Реализовать функцию, которая проверяет является ли массив отсортированным.

Адаптеры в Android

Теория. Адаптеры. Для чего нужны адаптеры. Готовые адаптеры в

Android: SimpleAdapter, ArrayAdapter. Абстрактный класс BaseAdapter.

Применение адаптеровдля обработкисобытийпользовательскогоинтерфейса.

Отображение ListView через адаптер. Методы getView(), getListAdapter().

Практика. Упражнение 5.2.1. Разбор примера кода с реализацией ListView черезArrayAdapter.

Упражнение 5.2.2. С помощью SimpleAdapter реализовать более сложный список, в котором строка формируется из двух текстов.

Ассоциативные массивы

Теория. 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. Идеяконтейнера: поиск ключей с помощью хэш-кодов значений.

Контейнер TreeMap. Контейнер TreeMap, хранениеданных, упорядоченных поключу. 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.

Практика. Упражнение 5.3.1. Разобрать пример использования TreeMap и HashMap.

Задание 5.3.1. 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.

Реляционная модель данных. СУБД. Введение в SQL

Теория. 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.

Локальная СУБД SQLite. Знакомство с локальной СУБД SQLite.

Практика. Минипроект 5.4. Записная книжка. Создать БД SQLite «Записная книжка» поспроектированной 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 ALTERTABLE 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 фильтр.

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

Практикум. Контрольное тестирование

Практика. Практическое занятие по темам раздела. Тестирование по всем темам раздела направленное на оценку практических знаний и навыков.

Раздел 5. Индивидуальное проектирование

Теория. Консультация по вопросам индивидуального проекта

Практика. Презентация и защита итогового проекта

1.5. 1.5. Формы контроля /аттестации и его периодичность

Система контроля знаний и умений обучающихся представляется в виде учёта результатов по итогам выполнения контрольных тестов по модулям. Метод педагогического наблюдения помогает отслеживать динамику развития обучающегося. В конце обучения подростки проходят защиту индивидуальных/групповых проектов.

Итоговый проект оценивается формируемой комиссией по 10-бальной шкале. Состав комиссии (не менее 3 человек): в обязательном порядке входит педагог; приветствуется привлечение ИТ-профессионалов, представителей высших и других учебных заведений, администрации учебной организации.

Компонентами оценки индивидуального проекта являются (по мере убывания значимости): качество ИП, отзыв руководителя проекта, уровень презентации и защиты проекта. Если проект выполнен группой обучающихся, то при оценивании учитывается не только уровень

исполнения проекта в целом, но и личный вклад каждого из авторов. Решение принимается коллегиально.

Формы проведения итогов по каждой теме и каждому разделу общеразвивающей программы соответствуют целям и задачам ДООП.

Раздел №2. «Комплекс организационно-педагогических условий»

2.1. Методическое обеспечение

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).

Программой предусмотрены следующие виды деятельности обучающихся:

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

Выбор методов обучения осуществляется исходя из анализа уровня готовности обучающихся к освоению содержания модуля, степени сложности материала, типа учебного занятия.

Формы обучения:

- фронтальная – предполагает работу педагога сразу со всеми обучающимися в едином темпе и с общими задачами. Для реализации обучения используется компьютер педагога с мультимедиа проектором, посредством которых учебный материал демонстрируется на общийэкран. Активно используются Интернет-ресурсы;
- групповая – предполагает, что занятия проводятся с подгруппой. Для этого группа распределяется на подгруппы не более 6 человек, работа в которых регулируется педагогом;
- индивидуальная – подразумевает взаимодействие преподавателя с одним обучающимся. Как правило данная форма используется в сочетании с фронтальной. Часть занятия (объяснение новой темы) проводится фронтально, затем обучающийся выполняют индивидуальные задания или общие задания в индивидуальном темпе.

Формы организации учебного занятия:

В образовательном процессе помимо традиционного учебного занятия используются многообразные формы, которые несут учебную нагрузку и могут использоваться как активные способы освоения детьми образовательной программы, в соответствии с возрастом обучающихся, составом группы, содержанием учебного модуля: беседа, лекция, мастер-класс, практическое занятие, защита проектов, конкурс, викторина, диспут, круглый стол, «мозговой штурм», воркшоп, квиз.

Некоторые формы проведения занятий могут объединять несколько

учебных групп или весь состав объединения, например, экскурсия, викторина, конкурс и т. д.

Дидактические материалы:

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;
- дополнительный учебный материал по теме;
- инструкции по настройке среды разработки.

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 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.

2.2. Условия реализации программы

Материально-техническое обеспечение:

Первый модуль программы реализуется организацией – участником в соответствии с условиями договора о сетевой форме реализации программ.

Требования к помещению:

– помещение для занятий, отвечающие требованиям СанПин для учреждений дополнительного образования;

– качественное освещение;

– столы, стулья по количеству обучающихся и 1 рабочим местом для педагога;

Оборудование:

– компьютеры и ноутбуки на каждого обучающегося и преподавателя;

– проекционное оборудование – 2 шт.

– маркерная доска – 1 шт.

Расходные материалы:

– whiteboard маркеры;

– бумага писчая;

– шариковые ручки;

– permanent маркеры.

Информационное обеспечение:

– операционная система (желательно Windows);

– программное обеспечение Eclips, Android Studio, объединенные в локальную сеть;

– планшет (для отладки);

– ПК для педагога, объединённый с функцией сервера.

Кадровое обеспечение:

Реализовывать программу может педагог, имеющий среднее специальное или высшее педагогическое образование по специальностям технического профиля, обладающий достаточными знаниями и опытом практической работы с подростками и получивший дополнительное образование (курсы повышения квалификации) в области программирования в таких средах, как AppInventor, AndroidStudio, а также со знаниями языка программирования Java.

Формы аттестации и оценочные материалы

Система контроля знаний и умений обучающихся представляется в виде учёта результатов по итогам выполнения контрольных тестов по модулям. Метод педагогического наблюдения помогает отслеживать

динамику развития обучающегося. В конце обучения подростки проходят защиту индивидуальных/групповых проектов.

Итоговый проект оценивается формируемой комиссией по 10-бальной шкале. Состав комиссии (не менее 3 человек): в обязательном порядке входит педагог; приветствуется привлечение ИТ-профессионалов, представителей высших и других учебных заведений, администрации учебной организации.

Компонентами оценки индивидуального проекта являются (по мере убывания значимости): качество ИП, отзыв руководителя проекта, уровень презентации и защиты проекта. Если проект выполнен группой обучающихся, то при оценивании учитывается не только уровень исполнения проекта в целом, но и личный вклад каждого из авторов. Решение принимается коллегиально.

Формы проведения итогов по каждой теме и каждому разделу общеразвивающей программы соответствуют целям и задачам ДООП.

Минимальное количество баллов, которое возможно получить по результатам промежуточного контроля по первому модулю - 1 балл, максимальное – 10 баллов.

Минимальное количество баллов, которое возможно получить по результатам промежуточного контроля по второму модулю - 1 балл, максимальное – 14 баллов.

Минимальное количество баллов, которое возможно получить по
результатам промежуточного контроля по третьему модулю - 1 балл, максимальное – 11 баллов.

Минимальное количество баллов, которое возможно получить по результатам промежуточного контроля по четвертому модулю - 1 балл, максимальное – 14 баллов.

Минимальное количество баллов, которое возможно получить по результатам защиты годового проекта - 1 балл, максимальное – 50 баллов.

Сумма баллов результатов промежуточного контроля и защиты итогового годового проекта переводится в один из уровней освоения образовательной программы согласно таблице 4:

Таблица 4

Баллы, набранные учащимся.	Уровень освоения
1-39	Низкий
40-79	Средний
80-100	Высокий

2.3. Календарный учебный график дополнительной общеобразовательной общеразвивающей программы «Мобильная разработка» (стартовый уровень)

№ п/п	Месяц	Число	Время проведения	Форма занятия	Кол-во часов	Тема занятия	Место проведения	Форма контроля
					32	***Основы программирования***	Учебный кабинет Компьютерный класс https://telemost.yandex.ru/	
1-4				Практикум	4	Вводное занятие. Инструктаж по ТБ. Среда разработки		Практическое задание, тестирование
5-12				Практикум	8	Арифметика. Примитивные типыданных. Логика. Операции отношения и логическиеоперации		Практическое задание, тестирование
13-16				Практикум	4	Условные конструкции. Блоки		Практическое задание, тестирование
17-24				Практикум	8	Итеративные конструкции. Массивы. Списки.		Практическое задание, тестирование
25-30				Практикум	6	Методы (функции). Видимость переменных.Рекурсия.		Практическое задание, тестирование
31-32				Практикум	2	Практикум. Контрольное тестирование		Практическое задание, тестирование
					38	***Объектно-ориентированное программирование***		
33-38				Практикум	6	Классы и объекты		Практическое задание, тестирование

39-44				Практикум	6	Классы: конструкторы, статические методы. Начальные приёмы тестирования и отладки		Практическое задание, тестирование
45-60				Практикум	16	Android. Структура. Активности. Интерфейспользователя. Язык разметки XML.ООП.		Практическое задание, тестирование
61-68				Практикум	8	Намерения. Фрагменты.		Практическое задание, тестирование
69-70				Практикум	2	Практикум. Контрольное тестирование.		Практическое задание, тестирование
					34	***Основы программирования Android-приложений***		
71-76				Практикум	6	Ввод, вывод и исключения		Практическое задание, тестирование
77-82				Практикум	6	Внутренние классы в обработке событий		Практическое задание, тестирование
83-88				Практикум	6	Параллелизм и синхронизация. Потоки		Практическое задание, тестирование
89-94					6	Двумерная графика в Android-приложениях		Практическое задание, тестирование
95-102					8	Реализация графики наоснове SurfaceView		Практическое задание, тестирование
103-104					2	Практикум. Контрольное тестирование		Практическое задание, тестирование

					30	***Алгоритмы и структуры данных***		
105 - 112					8	Массивы. Списки. Алгоритмы сортировки.Алгоритм двоичного поиска. Деревья		Практическое задание, тестирование
113 - 118					6	Адаптеры в Android		Практическое задание, тестирование
119 - 124					6	Ассоциативные массивы		Практическое задание, тестирование
125 - 132					8	Реляционная модель данных. СУБД. Введение в SQL		Практическое задание, тестирование
133 - 134					2	Практикум. Контрольное тестирование		Практическое задание, тестирование
135 - 144					***10***	***Индивидуальное проектирование***		Проект
					144	***Итого***		

2.4. Оценочные материалы

Оценочные материалы образовательных результатов

Показатели (оцениваемые параметры)	**Степень выраженности оцениваемого качества**	**Число баллов**	**Методы диагностики**
Теоретические знания по разделам/темам учебно-тематического плана программы	овладел менее чем ½ объема знаний, предусмотренных программой	1	Наблюдение, тестирование, защита работы и др.
	объем усвоенных знаний составляет более ½	2	
	освоил практически весь объем знаний, предусмотренных программой за конкретный период	3	
Практические умения и навыки, предусмотренные программой	овладел менее чем ½ предусмотренных умений и навыков	1	Наблюдение, защита работы
	объем усвоенных умений и навыков составляет более ½	2	
	овладел умениями и навыками, предусмотренными программой за конкретный период	3	

Оценочные материалы личностных результатов

Показатели (оцениваемые параметры)	**Степень выраженности оцениваемого качества**	**Число баллов**	**Методы диагностики**
Сформированность активности, организаторских способностей	мало активен, наблюдает за деятельностью других, забывает выполнить задание. Результативность невысокая	1	**Наблюдение**
	активен, проявляет стойкий познавательный интерес, трудолюбив, добивается хороших результатов	2	
	активен, проявляет стойкий познавательный интерес, добивается выдающихся результатов, инициативен, организует деятельность других	3	
Сформированность коммуникативных навыков, коллективизма	поддерживает контакты избирательно, чаще работает индивидуально, публично не выступает	1	**Наблюдение**
	вступает и поддерживает контакты, не вступает в конфликты, дружелюбен со всеми, по инициативе руководителя или группы выступает перед аудиторией	2	
	легко вступает и поддерживает контакты, разрешает конфликты, дружелюбен со всеми, инициативен, по собственному желанию успешно выступает перед аудиторией	3	
Сформированность ответственности, самостоятельности, дисциплинированности	неохотно выполняет поручения. Начинает работу, но часто не доводит ее до конца.	1	**Наблюдение**
	справляется с поручениями	2	

	и соблюдает правила поведения только пр и наличии контроля и требовательности преподавателя; выполняет поручения охотно, ответственно. Хорошо ведет себя независимо от наличия или отсутствия контроля, но не требует этого о т других		
	выполняет поручения охотно, ответственно, часто по собственному желанию, может привлечь других. Всегда дисциплинирован, везде соблюд ает правила поведения, требует того же от других	3	
Сформированность креативности, склонности к самостоятельному творчеству	может работать в проектно-исследовательской группе при постоянной поддержке и контроле. Способен принимать творческие решения, но в основном использует тра диционные способы	1	**Наблюд ение**
	может разработать свой творческий проект с помощью педагога. Способен на творческие решения, но в основном использует традиционные способы	2	
	высокий творческий потенциал. Самостоятел ьно выполняет работы. Находит нестандартн ые решения, новые способы выполнения заданий	3	

Оценочные материалы метапредметных результатов

Показатели (оцениваемые параметры)	**Степень выраженности оцениваемого качества**	**Число баллов**	**Методы диагности ки**
Понимать и принимать учебную задачу, сформулированную педагогом	овладел менее чем ½ объема задач, предусмотренных программой	1	Наблюден ие
	объем усвоенных задач составляет более ½	2	
	демонстрирует полное понимание, предусмотренных программой задач за конкретный период	3	
Планировать свои действия на отдельных этапах работы над выполнением творческого задания	овладел менее чем ½ объема знаний, предусмотренных программой	1	Наблюден ие
	демонстрирует неполное освоение планируемых действий, но более ½	2	
	освоил план действий в заданных условиях	3	
Осуществлять контроль, коррекцию и оценку результатов своей деятельности; понимать и применять полученную информацию при выполнении заданий	знает, но избегает их употреблять в деятельности	1	Наблюден ие
	демонстрирует неполное освоение заданных параметров, но более ½	2	
	освоил план действий в заданных условиях	3	

Мониторинг результатов обучения ребенка по дополнительной общеразвивающей программе

Показатели (оцениваемые параметры)	Методы диагностики
1. Уровни знаний / пониманий • Наличие общих представлений (менее ½ объема знаний) • Наличие ключевых понятий (объем усвоенных знаний более 1/2) • Наличие прочных системных знаний, (освоен практически весь объем)	Наблюдение, тестирование, контрольный опрос, собеседование
2. Уровни умения применять знания на практике • Репродуктивный несамостоятельный (деятельность осуществляется под непосредственным контролем преподавателя на основе устных и письменных инструкций). • Репродуктивный самостоятельный (деятельность осуществляется на основе типовых алгоритмов). • Творческий (в процессе деятельности творчески используются знаний, умений, предлагаются и реализуются оригинальные решения)	Контрольное задание
3. Наличие опыта самостоятельной деятельности • Очень незначительный опыт; • Незначительный балл (от случая к случаю); • Эпизодическая деятельность; • Периодическая деятельность; • Богатый опыт (систематическая деятельность)	Анализ, исследовательские работы, конкурсные работы, наблюдение
4. Сформированность личностных качеств • Очень низкая (проявились отдельные элементы); • Низкая (проявилась частично); • Недостаточно высокая (проявилась в основном); • Высокая (проявились полностью)	Анализ, наблюдение, собеседование

На основе вышеприведенного анализа заполняется диагностическая карта (оценочный лист) таблица 2.

Диагностическая карта успеваемости воспитанников объединения

Ф.И.О.	Знать / понимать (мах-3 балла)					Уметь использовать (мах-4 балла)					Владеть опытом (мах-5 баллов)					Личностные качества (мах-4 балла)					Итого баллов	Оценка
	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5		
Иванов А.																						

Результаты деятельности каждого обучающегося по каждому из показателей суммируются для определения итогового балла. Показатель усвоения (продуктивности обучения) вычисляется по формуле:

К усв = Ф/П *100%

Где К усв- коэффициент усвоения

Ф – фактический объем знаний (набранная сумма баллов)

П – полный объем знаний (максимальная сумма баллов).

В дальнейшем можно перейти к пятибалльной системе оценки.

Коэффициент сформированности:

80-100 - «отлично»
50-79 - «хорошо»
30-49 - «удовлетворительно»
Менее 29 - «неудовлетворительно»

Лист учета «Проект по мобильной разработке»

В помощь участникам проекта можно предложить заполнить следующий учётный лист.

Проект по мобильной разработке

Тема проекта:
Творческое название (при наличии):Основополагающий вопрос:
Авторы: 1. 2. 3. ... Предметная область:
Краткая аннотация:
Этапы выполнения проекта:

При подготовке к защите проекта учащимся необходимо подготовить презентацию и доклад, в котором отражаются основные этапы разработки программы, представлен алгоритм решения задачи, листинг программы, основные результаты работы. Можно предложить в помощь учащимся заполнить следующий чек-лист:

1. Аннотация.
2. Содержание.
3. Постановка задачи: a. Возможности использования программы b.Описание интерфейса
4. Формализация алгоритма: a. Перечень подпрограмм (при наличии) b. Описание алгоритма (блок-схема или подробное словесное описание алгоритма)
5. Листинг программы (текст программы).
6. Тестовые примеры a. Результаты работы b. Скриншотырезультатов работы
7. Описание размещения.
8. Требования к программным и аппаратным средствам.
9. Для оценивания проекта могут быть разработаны специальныеоценочные листы.

Ниже представлен пример оценочного листа:

Лист оценивания проекта

Критерий оценивания	*Группа 1*	*Группа 2*	*…*
Актуальность темы			
Соответствие содержания проекта			

заявленной теме			
Техническая сложность разработанной программы			
Оригинальность алгоритма			
Дизайн интерфейса			
Степень разработанности программы			
Применение программы для решения аналогичных задач			
Итоговое количество баллов			

2.5. Список литературы

Литература для обучающихся:

1. Брайсон, П. Легкий способ выучить Java /Пейн Брайсон – М.: БОМБОРА, 2019. – 400 с.
2. Корягин, А.В. Играй, программируй и создавай свои миры / А.В. Корягин - СПб.:Питер, 2021. - 240 с.
3. Уитни, Д. Программирование для детей. Учимся создавать сайты, приложения и игры.HTML, CSS и JavaScript / Д. Уитни; Пер. И. Рузмайкина – СПб. : Питер, 2018. – 208 с.
4. Файн, Я. Программирование на Java для детей, родителей, бабушек и дедушек / ЯковФайн - СПб.: Питер, 2011. - 231 с.
5. Федотенко, М.А. Разработка мобильных приложений. Первые шаги / М.А. Федотенко -М. : Лаборатория знаний, 2019. - 336 с.

Литература для педагога:

Общепедагогическая, психологическая и методическая литература

6. Гин, А.А. Приѐмы педагогической техники: свобода выбора, открытость, деятельность, обратная связь, идеальность: Пособие для учителей / А.А. Гин. – Гомель : ИПП «Сож», 1999. – 88 с.
7. Григорьев, Д.В. Внеурочная деятельность школьников. Методический конструктор: пособие для учителя / Д.В. Григорьев, П.В. Степанов. – М. : Просвещение, 2011. – 223 с. – (Стандарты второго поколения).
8. Леонтович, А.В. Проектная мастерская 5-9 классы. Учебное пособие. ФГОС./ А.В. Леонтович, А.С. Саввичев, И.А. Смирнов //. – М. : Просвещение, 2021. - 112 с. –(Внеурочная деятельность).
9. Найниш, Л.А. Инженерная педагогика: Научно-методическоепособие / Л.А. Найниш, В.Н. Люсев–М. : ИНФРА-М, 2019. - 88 с.

Специальная литература по теории и практике мобильных разработок

1. AITech - Using Procedures and Any component blocks (на англ.языке) [Электронный ресурс] URL: https://appinventor.mit.edu/explore/blogs/karen/2016/07-0.html (дата обращения: 19.03.2021).
2. База данных TinyDB (на англ.языке) [Электронный ресурс] URL: https://tinydb.readthedocs.io/en/latest/ (дата обращения: 19.03.2021).
3. Бэйтс, Б. Изучаем Java / Б. Бэйтс, К. Сьерра – М. : Эксмо, 2012. – 720 с.
4. Гриффитс, Д. Head First. Программирование для Android / Дэвид Гриффитс, ДонГриффитс; Пер. Е. Матвеев, Н. Римицан - СПб. : Питер, 2018. – 912 с.
5. Дарвин, Я.Ф. Android. Сборник рецептов. Задачи и решения для разработчиковприложений / Ян Ф. Дарвин – М. : Вильямс, 2017. – 768 с.
6. Дейтел, П. Android для разработчиков / П. Дейтел, Х. Дейтел, А. Уолд; 3-е издание —СПб. : Питер, 2016. – 496 с.
7. Игра «Найди золото» (на англ.языке) [Электронный ресурс] URL: https://drive.google.com/drive/folders/1xRSZGMLmtU7nJn22ToWCZIC92Z_bPaE(дата обращения:19.03.2021).
8. Игра Пианино (на англ.языке) [Электронный ресурс] URL: https://drive.google.com/drive/folders/1f9D_bQPy-G17EmdPCpY3-KoKAfH1E7qE (дата обращения: 19.03.2021).
9. Марсикано, К. Android. Программирование для профессионалов/ Кристин Марсикано,К. Стюарт, Билл Филлипс - СПб. : Питер, 2017. – 688 с.
10. Процедуров, А.И. (на англ.языке) [Электронный ресурс] URL: https://appinventor.mit.edu/explore/ai2/support/concepts/procedures (дата обращения: 19.03.2021).
11. Установка эмулятора (на англ.языке) [Электронный ресурс] URL:http:// appinventor.mit.edu/explore/ai2/setup-emulator (дата обращения: 19.03.2021).
12. Установка эмулятора в ОС Windows (на англ.языке) [Электронный ресурс] URL:http://appinventor.mit.edu/explore/ai2/windows (дата обращения: 19.03.2021).
13. Язык Kawa (на англ.языке) [Электронный ресурс] URL: https://www.gnu.org/ software/kawa/index.html (дата обращения: 19.03.2021).

Электронные ресурсы:

Портал обучения (https://informatics.msk.ru/)

Портал разработчика Android (https://developer.android.com/)