Государственное автономное нетиповое образовательное учреждение
Свердловской области «Дворец молодёжи»
Центр цифрового образования детей «IT-куб» «Солнечный»

Принята на заседании
научно-методического совета
ГАНОУ СО «Дворец молодёжи»
Протокол № 4 от 03 июня 2021 г.

УТВЕРЖДАЮ:
Директор
ГАНОУ СО «Дворец молодёжи»
А.Н. Слизько
Приказ № 464-д от 04 июня 2021 г.

ДВОРЕЦ МОЛОДЁЖИ
ДОКУМЕНТ ПОДПИСАН
ЭЛЕКТРОННОЙ ПОДПИСЬЮ
Сертификат: 02487D860079AC25AD4017CA3416FD0841
Владелец: Слизько Александр Николаевич
Действителен: с 20.11.2020 до 08.01.2022

Дополнительная общеобразовательная общеразвивающая программа
технической направленности

«Мобильная разработка»

Стартовый уровень

Возраст обучающихся: 14–17 лет
Срок реализации: 1 года

СОГЛАСОВАНО:
Начальник центра цифрового
образования детей «IT-куб»
«Солнечный» Пермяков А.В.
«19» мая 2021 г.

Авторы-составители:
Хомутов А.И.
Шмелев А.А.
педагоги дополнительного
образования,
Галицких К.В., заместитель
начальника по
образовательной
деятельности,
Бирюкова Е.А., методист,
Козлова А.А., методист

г. Екатеринбург, 2021 г.

I. Комплекс основных характеристик программы

1. Пояснительная записка

С целью создания равных условий получения качественного дополнительного образования, в рамках национального проекта «Образование» открываются центры цифрового образования детей «IT-куб». Эти центры образования детей по программам, направленным на ускоренное освоение актуальных и востребованных знаний, навыков и компетенций в сфере информационных технологий. Проект формирует современную образовательную экосистему, объединяющую компании-лидеров ИТ-рынка, опытных наставников и начинающих разработчиков.

На сегодня разработка программного обеспечения является одним из наиболее востребованных направлений в различных сферах экономики. Разработка ПО существует и как самостоятельная индустрия в области ИТ, и как сопутствующая во многих других сферах экономики.

В свою очередь стремительный рост мобильных платформ ещё далёк от своего предела. Кроме того, операционные системы, изначально предназначенные для мобильных персональных устройств, активно внедряются на новые типы устройств и машины.

Наиболее популярной такой мобильной операционной системой является ОС Android, устанавливаемая на совершенно разные платформы, от автомобилей до «умных плат», часов, телевизоров и различных приставок.

С другой стороны, ОС Android поддерживает приложения на популярном языке Java, на котором создана большая часть приложений платформы. В современном мире Java как платформа является наиболее популярной в связи с тем, что не имеет требований к операционной системе для запуска своих приложений.

Изучение языка программирования Java по данной программе обучения даёт возможность пользователю мобильного устройства с ОС Android создавать программы, взаимодействующие с элементами графики, аудио и видеофайлами, текстовыми форматами. Кроме того, освоив основы языка Java, обучающийся

получает возможность достаточно быстро мигрировать в область разработки серверных и десктоп-приложений на платформе Java, что значительно увеличивает универсальность и востребованность специалиста.

Направленность программы – техническая.

Ориентирована на развитие навыков программирования и проектирования программ под платформу Android.

Данная общеразвивающая программа основывается на следующих нормативных документах:

- Федеральным законом от 29 декабря 2012 г. № 273-ФЗ «Об образовании в Российской Федерации»;
- Приказа Министерства образования и науки Российской Федерации от 29 августа 2013 г. № 1008 «Об утверждении порядка организации и осуществления образовательной деятельности по дополнительным общеобразовательным программам»;
- «Санитарно-эпидемиологических требований к условиям и организации обучения в общеобразовательных учреждениях», утвержденных Главным государственным санитарным врачом РФ 29 декабря 2012 года № 189;
- Письма Минобрнауки РФ от 11.12.2006 N 06-1844 «О примерных требованиях к программам дополнительного образования обучающихся»

Актуальность программы обусловлена потребностью общества в технически грамотных специалистах и полностью отвечает социальному заказу по подготовке квалифицированных кадров в области мобильной разработки.

Прогностичность программы «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.

Отличительная особенность программы «Мобильная разработка» – в изучении основ языка программирования Java и структуры приложения под ОС Android, а также в ознакомлении с наиболее востребованными паттернами программирования и нотации программного кода. 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.

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 средних специальных и высших учебных заведениях.

Адресат программы

Дополнительная общеразвивающая программа «Мобильная разработка» предназначена для детей в возрасте 14–17 лет, без ограничений возможностей здоровья. Количество обучающихся в группе – 10–14 человек. Состав групп постоянный.

Место проведения занятий: г. Екатеринбург, ул. Чемпионов 11.

Возрастные особенности групп

Выделенные нами возрастные периоды при формировании групп:

– *14 лет* – подростковый период. Характерная особенность – личное самосознание, сознательное проявление индивидуальности. Ведущая потребность – самоутверждение. В подростковый период стабилизируются интересы детей. Основное новообразование – становление взрослости как стремление к жизни в обществе взрослых. К основным ориентирам взросления относятся: социально-моральные – наличие собственных взглядов, оценок, стремление их отстаивать; интеллектуально-деятельностные – освоение элементов самообразования, желание разобраться в интересующих подростка областях; культурологические – потребность отразить взрослость во внешнем облике, манерах поведения. Роль педагога дополнительного образования в работе с подростками заключается в том, чтобы регулярно осуществлять их подготовку к самопрезентации социально значимой группе людей.

– *15–17 лет* – юношеский возраст. Завершение физического и психического созревания. Социальная готовность к общественно полезному производительному труду и гражданской ответственности. В отличие от подросткового возраста, где проявление индивидуальности осуществляется благодаря самоидентификации –

«кто я», в юношеском возрасте индивидуальность выражается через самопроявление – «как я влияю». Основная задача педагога дополнительного образования в работе с детьми в возрасте 15–16 лет сводится к решению противоречия между готовностью их к полноценной социальной жизни и недопущением отставания от жизни содержания и организации их образовательной деятельности.

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.

Режим занятий и объём общеразвивающей программы: длительность одного занятия составляет 2 академических часа, периодичность занятий – 2 раза в неделю.

Срок освоения общеразвивающей программы определяется содержанием программы и составляет 1 год (144 часа).

Формы обучения и виды занятий.

Сочетание очной и очно-заочной форм образования с применением электронного обучения и дистанционных образовательных технологий (Закон №273-ФЗ, гл.2, ст.17, п.2.).

Учебный процесс строится таким образом, чтобы экспериментальная и практическая работа преобладала над теоретической подготовкой. Необходимые для работы теоретические сведения предоставляются педагогом перед началом практических занятий в общем доступе. Индивидуальная работа проводится во время практических занятий. Групповая работа проводится во время теоретических занятий. Каждая тема по программированию сопровождается раскрытием 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\допустимы следующие виды деятельности обучающихся:

- освоение теоретического и практического материала на занятиях;
- разработка индивидуального проекта;
- участие в вебинарах;
- промежуточная аттестация в форме 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.

Объём общеразвивающей программы: 144 часа.

Форма организации образовательной деятельности – групповая.

Программа является одногодичной, модульной. Каждый модуль направлен на формирование определённых компетенций.

Модуль – структурная единица образовательной программы, имеющая определённую логическую завершённость по отношению к результатам обучения. 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. Возможно выполнение сквозного «продукта», тогда результатом освоения модуля будет соответствующий законченный функционал «продукта».

Кейс – история, описывающая реальную ситуацию, которая требует проведения анализа, выработки и принятия обоснованных решений.

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».

Педагогическая целесообразность программы

Программа реализует профориентационные задачи, обеспечивает знакомство с современными профессиями в сфере IT.

Продвинутый уровень предполагает использование форм организации материала, обеспечивающих доступ к сложным и нетривиальным разделам в рамках содержательно-тематического направления общеразвивающей программы. Также предполагает углублённое изучение содержания общеразвивающей программы и доступ к околопрофессиональным и профессиональным знаниям в рамках содержательно-тематического направления общеразвивающей программы.

Осваивая данную программу, обучающиеся будут овладевать навыками востребованных на рынке труда. Практически для каждой перспективной профессии будут полезны знания и навыки, рассматриваемые в данной программе. Программа помогает решать проблемы личностного и профессионального самоопределения, самореализации подростков.

2. Цели и задачи программы

Цель программы: овладение базовыми навыками и теоретическими основами проектирования приложений под современную платформу Android, формирование технической грамотности средствами приобщения обучающихся к разработке программ.

Для успешной реализации поставленной цели необходимо решить ряд педагогических, развивающих и воспитательных задач:

Обучающие:

- 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 для Android;
- обучение программированию технических устройств;
- знакомство с современными нотациями программирования и некоторыми шаблонами программирования

Развивающие:

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

- воспитание этики групповой работы, отношений делового сотрудничества, взаимоуважения;

- развитие основ коммуникативных отношений внутри проектных групп и в коллективе в целом;
- воспитание упорства в достижении результата;
- формирование целеустремлённости, организованности, ответственного отношения к труду, толерантности и уважительного отношения к окружающим.

3. Содержание общеразвивающей программы

Учебный план

№ пп	Наименование раздела, темы	Количество часов			Формы аттестации / контроля
		Теория	Практика	Всего	
	Входной контроль, оценка знаний				
	Модуль 1. Логика	**4**	**2**	**6**	
1.1	ТБ. «Событие». Введение в алгебру логики. Элементы алгебры логики.	1	1	2	Устный опрос. Инструктаж по ТБ и ПБ
1.2	Логические операции. Таблицы истинности. Решение логических задач.	1	1	2	Устный опрос
1.3	Множества. Системы счисления. Представление данных в компьютере. Алгоритмы и блок-схемы.	1	1	2	Устный опрос
	Модуль 2. Основы программирования	**9**	**16**	**26**	
2.1	Среда разработки. Примитивные типы данных. Арифметика в информатике.	1	1	2	Устный опрос
2.2	Тип boolean. Операции отношения и логические операции. Приведение типов.	1	1	2	Устный опрос
2.3	Условные конструкции. Блоки.	1	1	2	Устный опрос
2.4	Итеративные конструкции	2	2	4	Устный опрос
2.5	Массивы	1	1	2	Устный опрос
2.6	Многомерные и неровные массивы	1	1	2	Устный опрос
2.7	Строки	1	1	2	Устный опрос
2.8	Методы (функции). Видимость переменных	2	2	4	Устный опрос
2.9	Практикум		4	4	Устный опрос
2.10	Контрольное тестирование		2	2	Тест
	Модуль 3. Объектно-ориентированное программирование	**8**	**18**	**26**	
3.1	Классы и объекты, описание	1	1	2	Устный опрос

	класса.				
3.2	Инкапсуляция	1	1	2	Устный опрос
3.3	Наследование	2	2	4	Устный опрос
3.4	Полиморфизм	2	2	4	Устный опрос
3.5	Классы: конструкторы, статические методы	2	2	4	Устный опрос
3.6	Практикум		8	8	Визуальный контроль
3.7	Контрольное тестирование по модулю		2	2	Тест
	Модуль 4. Основы программирования Android-приложений	**7**	**13**	**20**	
4.1	IDE Android Studio. Структура проекта и архитектура приложений под Android	1	1	2	Устный опрос
4.2	Интерфейс пользователя. Язык разметки XML	1	1	2	Устный опрос
4.3	Намерения (Intents), Context. Внутренние классы	1	1	2	Устный опрос
4.4	Модель обработки событий. Шаблон Listener\Слушатель. Внутренние классы в обработке событий	2	2	4	Устный опрос
4.5	Элементы GUI (основные элементы)	2	2	4	Устный опрос
4.6	Практикум		4	4	Визуальный контроль
4.7	Контрольное тестирование по модулю		2	2	Тест
	Модуль 5. Алгоритмы и структуры данных. Базы данных.	**7**	**23**	**30**	
5.1	Элементы GUI. Контейнеры списочных данных.	1	1	2	Устный опрос
5.2	Списки, деревья, ассоциативные массивы, хэш-таблицы.	1	1	2	Устный опрос
5.3	Представление списочных данных на экране. Адаптеры.	2	2	4	Устный опрос
5.4	Базы данных, СУБД. Реляционная модель данных.	1	3	4	Устный опрос
5.5	Введение в SQL, инструкции DDL	1	3	4	Устный опрос
5.6	Введение в SQL, SQL инструкции DML	1	3	4	Устный опрос
пр	Практикум		8	8	Визуальный контроль

к	Контрольное тестирование по модулю.		2	2	Тест
	Модуль 6. Основы разработки сетевых приложений, параллельного выполнения и контроль ошибок	**11**	**25**	**36**	
6.1	IP-сети, и архитектура сетевого взаимодействия.	1	1	2	Устный опрос
6.2	Работа с сетью в Android.	2	2	4	Устный опрос
6.3	Ввод\вывод	2	2	4	Устный опрос
6.4	Исключения. Обработка исключений	2	2	4	Устный опрос
6.5	Параллелизм и синхронизация. Потоки	2	2	4	Устный опрос
6.6	Планировщик заданий. Сервисы в Android	2	4	6	Устный опрос
6.7	Практикум		8	8	Визуальный контроль
6.8	Контрольное тестирование по модулю		2	2	Тест
	Итоговая защита		2	2	
	Итого	**46**	**97**	**144**	

Содержание учебного плана

Модуль 1. Логика

Тема 1.1. 2 часа. ТБ. «Событие». Введение в алгебру логики. Элементы алгебры логики.

Тема 1.2. 2 часа. Логические операции. Таблицы истинности. Решение логических задач.

Тема 1.3. 2 часа. Множества. Системы счисления. Представление данных в компьютере. Алгоритмы и блок-схемы.

Тема 1.1. ТБ. «Событие». Введение в алгебру логики. Элементы алгебры логики.

Цель. Сформировать принцип безопасной работы за компьютером в классе и дома. Познакомить с алгеброй логики, основными понятиями.

Алгебра логики – определение понятия, для чего изучается.

События и высказывания в алгебре логики. Истинные и ложные высказывания, простые и сложные.

Упражнение 1.1.1. Примеры высказываний. Примеры простых и сложных высказываний.

Задание 1.1.1. Составить простое истинное высказывание. Составить Сложное истинное высказывание.

Практика: Импликация (если…, то…), рассмотреть отношения между понятиями по типу:

Если А, то Б

Б в том случае, если А

При А будет Б

Из А следует Б

В случае А произойдет Б

Б, так как А

Б, потому что А

А – достаточное условие для Б

Б – необходимое условие для А

Тема 1.2. Логические операции. Таблицы истинности. Решение логических задач.

Цель. Изучить основные логические операции. Сформировать понимание таблиц истинности, для чего они применяются, как составлять. Принцип решения логических задач через логические операции и таблицы истинности.

Логическая операция – определение, суть понятия. Разобрать основные логические операции\функции: **инверсия, конъюнкция, дизъюнкция.**

Таблицы истинности для каждой операции.

Приоритет логических операций (формульные записи логических высказываний).

Сложные логические операции (**или-не, и-не, исключающее или, эквивалентность, импликация**) и выражение их через простые основные, через таблицы истинности.

Упражнение 1.2.1 Пример решения практических задач с помощью таблиц истинности.

Задание 1.2.1 Решение логических задач.

Тема 1.3. Множества. Системы счисления. Представление данных в компьютере. Алгоритмы и блок-схемы

Цель. Изучить понятие «множество», операции над множествами. Познакомиться с различными системами счисления. Познакомиться с принципом размещения данных в компьютере. Сформировать понятие алгоритмов вычислений и блок-схем алгоритмов.

Множество как объединение некоторого количества **элементов множества**. Формы обозначения и записи множеств и его элементов. **Пустое множество**.

Основные отношения множеств (равенство, включение\подмножество, пересечение, объединение).

Обзорное представление логических операции над множествами, их символьное и графическое обозначение. Обзорные доказательства операций над множествами с помощью диаграмм Эйлера-Венна.

Упражнение 1.3.1 Доказательство простых тождеств (на трёх множествах) с помощью диаграмм.

Число, цифра – определение и примеры, в чём существенная разница понятий. **Позиционные и непозиционные системы счисления** – определение и примеры.

Десятичная, двоичная, восьмеричная, шестнадцатеричная системы счисления – основная суть и форма записи, преимущества и недостатки, где используются.

Представление данных в компьютере – упрощённо как матрицу ячеек памяти. Кодирование любого вида информации в двоичных числах.

Алгоритм – определение, пример. Словесный алгоритм выполнения какой-либо работы. Алгоритм, как разбиение большой задачи на некую последовательность действий.

Упражнение 1.3.2 Составить словесный алгоритм какого-либо регулярного действия.

Задание 1.3.1 Составить словесный алгоритм по заданию учителя.

Графическое представление алгоритма. **Блок-схема** алгоритма. Описать графические блоки для записи алгоритма.

Упражнение 1.3.3 Составить блок-схему алгоритма по упражнению 1.3.2

Задание 1.3.2 Составить блок-схему алгоритма по заданию 1.3.1

Модуль 2. Основы программирования

Тема 2.1. 2 часа. Среда разработки. Примитивные типы данных. Арифметика в информатике.

Тема 2.2. 2 часа. Тип boolean. Операции отношения и логические операции. Приведение типов

Тема 2.3. 2 часа. Условные конструкции. Блоки.

Тема 2.4. 4 часа. Итеративные конструкции.

Тема 2.5 2 часа. Массивы.

Тема 2.6. 2 часа. Многомерные и неровные массивы.

Тема 2.7. 2 часа. Строки.

Тема 2.8. 4 часа. Методы (функции). Видимость переменных.

Практикум. 4 часа.

Контрольное тестирование по модулю. 2 часа.

Тема 2.1. Среда разработки. Примитивные типы данных. Арифметика в информатике.

Цель. Закрепление на практике базовых навыков работы со средой программирования и основ синтаксиса Java.

Рассмотреть примитивные типы данных, арифметические выражения и операторы, операторы присваивания, преобразования типов. Знакомство с Венгерской нотацией кода на примере переменных и примитивов.

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

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

Упражнение 2.1.1. 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.

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.

Нотация программного кода. Венгерская нотация, как общепринятый стандарт. Рекомендации по именованию переменных.

Упражнение 2.1.2. 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 2.1.3. 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 2.1.3. Написание простейших программ, демонстрирующих выполнение каждого изученного оператора. Объяснение результатов её выполнения.

Задание 2.1.1. 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 2.1.2. 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 2.1.3. 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 2.2. Тип Boolean. Операции отношения и логические операции. Приведение типов.

Цель. Рассмотреть операторы и их классификацию, поразрядные операции, логические выражения. Явное и неявное приведение типов.

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.

Явное и неявное приведение типов. Совместимые и несовместимые типы данных. Приведение типов данных при выполнении операций. Явное и неявное приведение типов (размерность типов при операциях приведения).

Упражнение 2.2.1. Программа, демонстрирующая выполнение логических операций и операций отношений. Объяснение результатов её выполнения.

Задание 2.2.1. 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 2.3. 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 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.

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 2.4. Итеративные конструкции

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

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 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.

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.

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 учителем формуле.

Тема 2.5. Массивы

Цель. Изучить понятие массивов, представление массива в памяти устройства.

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.

Преимущества и недостатки массивов вытекают из их представления в памяти компьютера (скорость доступа).

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

Задание 2.5.1. 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 2.6. 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. Пример неровных массивов (элементами массива могут быть массивы разной длинны). Ошибка обращения к несуществующему элементу массива.

Упражнение 2.6.1. 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. Ошибка обращения к несуществующему элементу массива.

Упражнение 2.6.2. Фрагменты кода, иллюстрирующие особенности объявления и обработки неровных массивов.

Задание 2.6.1. Написать программу поиска максимального/минимального числа, заданного числа:

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

Задание 2.6.2. Написать программу генерации игрового поля для головоломки Судоку размером 9x9. Написать программу решения заданной головоломки Судоку размером 9x9.

Тема 2.7. Строки

Цель. Изучить тип данных (**String**) для работы со строками. Сформировать понимание, о сложных типах данных – структурах данных. Показать, что строки не являются массивом символов.

Тип String для работы со строками в Java. Способ описания переменных, задание значений переменным. Операции со троками и строковыми переменными.

Пояснить что строковые переменные – это ссылочные переменные, значит некорректно сравнивать их значение через оператора равенства. Сравнение значений строк через специальный метод.

Пояснить, что строковые литералы создаются в памяти единожды, и различные переменные могут ссылаться на один литерал. Разница задания значения строковой переменной через оператор присвоения и оператор new(). Методы строковых переменных.

Упражнение 2.7.1 Продемонстрировать различные способы задания строковых переменных. Вывести результат сравнения этих переменных через оператор равенства и через специальный метод сравнения.

Тема 2.8. 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. Венгерская нотация именования функций и параметров.

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

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 2.8.1. 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 метода, область блока, обзорно для области класса и пакета.

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 3. Объектно-ориентированное программирование

Тема 3.1. 2 часа. Классы и объекты. Описание класса.

Тема 3.2. 2 часа. Инкапсуляция.

Тема 3.3. 4 часа. Наследование.

Тема 3.4. 4 часа. Полиморфизм.

Тема 3.5. 4 часа. Классы: конструкторы, статические методы.

Практикум. 8 часа.

Контрольное тестирование по модулю. 2 часа.

Тема 3.1. Классы и объекты. Описание класса

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

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, посылая 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.

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).

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

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 примеры объектов, полей и методов.

Тема 3.2. Инкапсуляция

Цель. Освоить понятие инкапсуляции и познакомиться с примерами его применения. Усвоить идею сокрытия внутренних данных объекта, и доступ к ним через методы объекта. Обзорно модификаторы доступа.

Раскрыть понятие и суть 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 должен инкапсулировать (содержать в себе) всё необходимое для совей работы, но не более того.

Упражнение 3.2.1. Проектирование и реализация простейшего класса. На примере скрыть поля. Предоставив доступ к ним через соответствующие методы. Объяснить для чего это нужно, какие могут возникнуть ошибки.

Автоматическая генерация getter/setter в классе с помощью среды разработки.

Задание 3.2.1. Придумать примеры классов и соответствующие им примеры объектов, полей и методов. Кие из этих полей следует скрыть, почему и каким образом?

Тема 3.3. Наследование

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». Множественное наследование интерфейсов.

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: открытый член класса для доступа из производных классов и закрытый для доступа извне.

Упражнение 3.3.1. 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.

Упражнение 3.3.2. Разобрать аналогичный пример, где одно из наследований заменено вложенностью классов. Объяснить различие в реализации.

Задание 3.3.1. В соответствии с заданным вариантом задания разработать собственный класс, аналог которого общеупотребителен в окружающем мире или в математике и имеет хорошо знакомое поведение

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 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).

Тема 3.4. 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 3.4.1. 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 3.4.1. Реализовать классы круг и овал, реализующее интерфейс Figure.

Тема 3.5. 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 3.5.1. 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 3.5.2. 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 3.5.1. 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.

Практикум

Практическое занятие по темам модуля. Конкретное содержание определяется учителем.

Контрольное тестирование по модулю

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

Модуль 4. Основы программирования Android приложений

Тема 4.1. 2 часа. IDE Android Studio. Структура проекта, запуск программы на эмуляторе. Архитектура приложений под Android. Активности.

Тема 4.2. 2 часа. Интерфейс пользователя. Язык разметки XML.

Тема 4.3. 2 часа. Намерения (Intents), Context. Внутренние классы

Тема 4.4. 4 часа. Модель обработки событий. Шаблон Listener\Слушатель. Внутренние классы в обработке событий.

Тема 4.5. 4 часа. Элементы GUI (основные элементы).

Практикум. 4 часа.

Контрольное тестирование по модулю. 2 часа.

Тема 4.1. IDE Android Studio. Структура проекта и архитектура приложений под Android.

Цель. Ознакомиться со средой разработки Android Studio. Эмуляторами андроид-устройтв. Изучить структуру андроид-проекта. Познакомиться с активностями (Activity) как основополагающим элементом программы.

Android Studio – интегрированная среда разработки, её возможности. Запуск и настройка эмуляторов устройств.

Структура проекта – основные разделы проекта, обязательные разделы.

Класс Activity в программе. Активность, как структурный элемент программы, класс, управляющий экраном. Создание активности.

Жизненный цикл андроид-приложения и отдельной активности. Раскрыть понятие жизненного цикла приложения. Жизненный цикл активности в контексте жизненного цикла приложения. Состояния активностей, и вызываемые методы при смене состояний.

Упражнение 4.1.1. Разбор простейшего андроид-проекта с одним пустым экраном.

Упражнение 4.1.2. Отслаивание жизненного цикла активити через методы onCreate(), OnStar(), onStop(), onResume(), onRestart(), onDestroy().

Задание 4.1.1. Модифицировать пример из упражнения таким образом, чтобы при смене состояний активности, сообщение о текущем состоянии выводилось на русском языке в лог.

Тема 4.2. 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 4.2.1. 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). Создание xml файла с описанием вида экрана в папке res/layout. Основные элементы интерфейса и их

описание в файле fragment_main.xml: EditText, TextView. Метода setContentView для загрузки xml файла из res/layout.

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

Упражнение 4.2.2. 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).

Задание 4.2.1. 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.).

Тема 4.3. Намерения (Intents), Context. Внутренние классы

Цель. Ознакомиться с классами Context, Intent как основополагающими механизмами приложения андроид.

Контекст (Context) Что такое Context и его использование. Контекст как интерфейс доступа к глобальной информации об окружении приложения. Виды контекстов (контекст приложения, Activity, базовый).

Намерения (Intents). Явные и неявные намерения. Намерения как механизм описания одной операции.

Создание нескольких Activity (и экранов) в одном приложении. Регистрация их в AndroidManifest.xml. Создание Intent для перехода на другой экран (метод startActivity).

Упражнение 4.3.1. Разработать 2 экрана, реализовать запуск второго экрана. Передать текстовое сообщение во второй экран. На втором экране проверить. Передана ли в него какая-либо информация.

Задание 4.3.1. Доработать упражнение, таким образом, чтобы в открывающимся окне переданное текстовое сообщение отображалось на экране в текстовой метке.

Тема 4.4. Модель обработки событий. Шаблон Listener\Слушатель. Внутренние классы в обработке событий

Цель. Изучить механизм обработки событий интерфейса в андроид-приложении. 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 4.4.1. Разбор примера кода с обработчиками событий.

Тема 4.5 Элементы GUI (основные элементы)

Цель. Ознакомиться с простыми компонентами пользовательского интерфейса.

Иерархия классов элементов GUI. Основные свойства, присущие всем элементам UI. Способы задания значений свойств элементов. Специфические свойства UI.

Упражнение 4.5.1 Составить макет экрана приложения с заданными свойствами отдельных элементов пользовательского интерфейса.

Задание 4.5.1 Доработать программу упражнения, таким образом, чтобы при нажатии на кнопку (добавить, если нужно) свойства некоторых элементов менялись на заданные учителем.

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. Алгоритмы и структуры данных. Базы данных.

Тема 5.1. 2 часа. Элементы GUI. Контейнеры списочных данных.

Тема 5.2. 2 часа. Списки, деревья, ассоциативные массивы, хэш-таблицы.

Тема 5.3. 4 часа. Представление списочных данных на экране. Адаптеры.

Тема 5.4. 4 часа. Базы данных. Системы управления БД. Типы СУБД. Реляционная модель данных.

Тема 5.5. 4 часа. Введение в SQL, инструкции DDL.

Тема 5.6. 4 часа. Введение в SQL, SQL инструкции DML.

Практикум. 8 часа.

Контрольное тестирование по модулю. 2 часа.

Тема 5.1 Элементы GUI. Контейнеры списочных данных

Цель. Изучить пользовательские элементы для отображения списочных данных.

Строковый массив в описании файла ресурсов string.xml, в каких случаях удобно использовать, и когда не подходит.

Упражнение 5.1.1. Описание строкового массива в фале string.xml.

Контейнер списков ListView. Назначение, особенности. Простота, использования. Возможность вывода значений из фала ресурсов.

Контейнер списков Spiner. Назначение, особенности, варианты использования.

Контейнер RecyclerView. Ознакомление с функционалом и особенностями по сравнению с ListView.

Задание 5.1.1 Вывести на экране в контейнере списков строковый массив из файла ресурсов.

Тема 5.2. Списки, деревья, ассоциативные массивы, хэш-таблицы

Цель. Освоение понятия «список» как структуры данных, изучение наиболее распространённых структур данных.

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).

Преимущества и недостатки массивов.

Класс ArrayList, его преимущества и недостатки по сравнению с обычными массивами. Примеры использования.

Класс HashMap, его особенности и варианты применения. Примеры использования

Упражнение 5.2.1. Пример применения классов ArrayList и HashMap.

Тема 5.3. Представление списочных данных на экране. Адаптеры

Цель. Освоить применение адаптеров для вывода списочных данных 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 5.4. Базы данных, СУБД. Реляционная модель данных

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 5.4. Записная книжка. Спроектировать БД для приложения «Записная книжка» на бумаге.

Тема 5.5. Введение в SQL, инструкции DDL

Цель. На примере локальной СУБД SQLite изучить средства SQL, используемые для создания структуры БД.

Локальная СУБД SQLite. Знакомство с локальной СУБД SQLite.

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.

Упражнение 5.5.1 Запрос создания таблицы БД из рассматриваемого примера.

Задание 5.5.1 Написать запросы создания остальных таблиц из рассматриваемого примера.

Тема 5.6. Введение в SQL, инструкции DML

Цель. На примере локальной СУБД SQLite изучить средства SQL, используемые для манипулирования данными.

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 ходе изучения Модуля.

Модуль 6. Основы разработки сетевых приложений, параллельного выполнения и контроль ошибок

Тема 6.1. 2 часа. IP-сети, и архитектура сетевого взаимодействия.

Тема 6.2. 4 часа. Работа с сетью в Android..

Тема 6.3. 4 часа. Ввод\вывод.

Тема 6.4. 4 часа. Исключения. Обработка исключений.

Тема 6.5. 4 часов. Параллелизм и синхронизация. Потоки.

Тема 6.6. 6 часов. Планировщик заданий. Сервисы в Android.

Практикум. 8 часов.

Контрольное тестирование по модулю. 2 часа.

Итоговая защита. 2 часа

Тема 6.1. IP-сети, и архитектура сетевого взаимодействия

Цель. Ознакомиться с общим устройством сетей, и IP-адресации. Получить представление об общей схеме передачи данных по сети. Получить общее представление о Web-сервере, структуре HTTP-запроса и ответа сервера, REST-взаимодействие.

Общие понятия. Интернет и протоколы TCP/IP. Адресация устройств в сетях: для чего она нужна. Маски подсети. Порты. 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. Доменные имена (DNS). URL-ссылки. Уяснить разницу между понятиями сайт, сервер, доменное имя.

Упражнение 6.1.1. 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.

Задание 6.1.1. Узнать собственный внешний IP-адрес на домашнем компьютере. Сравнить записанные значения в компьютерном классе со значениями дома. Объяснить, почему значения существенно отличаются.

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 6.1.1. 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.

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

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

Тема 6.2. Работа с сетью в Android

Цель. Получить практический навык организации сетевого взаимодействия в Andoid-приложениях.

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 6.2.1 Создание простой формы, которая отправляет значения двух полей «Имя» и «Фамилия» на сервер по кнопке «Отправить». 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 6.2.2. Модифицировать программу из упражнения 6.2.1 с использованием JSON и библиотеки Retrofit.

Тема 6.3. Ввод\вывод

Цель. Изучить файловый ввод вывод и механизм обработки исключений в Java.

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

Задание 6.3.1. 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 6.4. Исключения. Обработка исключений

Цель. Ошибки исполнения программ – термин «исключение». Изучить 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.

Упражнение 6.4.1 Доработать программу упражнения 6.3.1, так, чтобы обработать возможные исключения (ошибки ввода\вывода).

Тема 6.5. 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 6.5.1. Разобрать пример программы, совершающей загрузку картинки из интернета и устанавливающей её на экран.

Тема 6.5. Планировщик заданий. Сервисы в Android

Цель. Ознакомиться со стандартными возможностями SDK Android установления\планирования заданий на выполнения кода. Освоить практическое применение класса AlarmManager для автоматического запуска задачи. Ознакомиться с понятием Service в Андроид. Приобрести практический навык запуска сервисов, разработка собственного сервиса.

Класс AlarmManager как механизм запуска задания в определённое время и периодичностью. Методы класса **AlarmManager.**

Класс PendingIntent – применение.

Класс JobSheduler – класс планировщика задач. Отличительные особенности от AlarmManager.

Понятие сервисов в Андроид. Особенности сервисов (работают независимо от приложения в фоновом режиме).

Жизненный цикл сервиса и создание, запуск и остановка сервисов.

Класс IntentService, позволяющий запускать сервис в отдельном потоке.

Упражнение 6.5.1 Разработать приложение, которое в заданное время отправляет уведомление (типа Toast).

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 ходе изучения Модуля.

4. Планируемые результаты

Предметные результаты:

- 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 обучению, 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:

- 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;
- формирование и развитие компетентности в области использования информационно-коммуникационных технологий (ИКТ-компетенция);
- умение организовывать учебное сотрудничество и совместную деятельность с педагогом и сверстниками в процессе проектной и учебно-исследовательской деятельности.

II. Комплекс организационно-педагогических условий реализации общеразвивающей программы

1. Календарный учебный график на 2021-2022 учебный год

Таблица 2

№ п/п	Основные характеристики образовательного процесса	
1	Количество учебных недель	34
2	Количество учебных дней	68
3	Количество часов в неделю	4
4	Количество часов	144
5	Недель в I полугодии	15
6	Недель во II полугодии	19
7	Начало занятий	13 сентября
8	Каникулы	25 октября – 31 октября
9	Выходные дни	31 декабря – 9 января
10	Окончание учебного года	31 мая

2. Условия реализации программы

Материально-техническое обеспечение:

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;
- шкаф для оборудования.

Оборудование*:*

- ноутбук преподавателя HP Pavilion Gaming laptop 17 в комплекте;
- ноутбук обучающегося Lenovo v340-17iwl в комплекте с мышью – 12 шт;
- Интерактивная панель smart vision DC75-E4c подставкой;
- Wi-fi роутер keenetic Ultra;

Магнитно-маркерная доска – 1шт

Расходные материалы:

- whiteboard маркеры;
- бумага;
- шариковые ручки;
- permanent маркеры.

Информационное обеспечение:

- операционная система (Windows);
- программное обеспечение Eclips, Android Studio,
- планшет\мобильный телефон (для отладки);
- ПК для педагога.

Кадровое обеспечение

Программа реализуется Хомутовым А.И. и Шмелевым А.А., педагогами дополнительного образования.

При реализации программы другим педагогом стоит учитывать, что преподавателю необходимо познакомиться с технологией обучения по направлению «Мобильная разработка».

3. Формы аттестации и оценочные материалы

Система контроля знаний и умений учащихся представляется в виде учёта результатов по итогам выполнения заданий отдельных кейсов и посредством наблюдения, отслеживания динамики развития учащегося.

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

Баллы, набранные учащимся	**Уровень освоения**
0–50 баллов	Низкий
50–75 баллов	Средний
75–100 баллов	Высокий

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

Индивидуальный/групповой проект оценивается формируемой комиссией. Состав комиссии (не менее 3-х человек): педагог (в обязательном порядке), администрация учебной организации, приветствуется привлечение IT-профессионалов, представителей высших и других учебных заведений.

Компонентами оценки индивидуального/группового 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. Для оценки проекта членам комиссии рекомендуется использовать «Бланк оценки ИП» (Приложение 5).

4. Методические материалы

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 минисоревнований).
- 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 общий экран. 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;
- ***дистанционная*** – взаимодействие педагога и обучающихся между собой на расстоянии, отражающее все присущие учебному процессу компоненты. Для

реализации дистанционной формы обучения весь дидактический материал размещается в свободном доступе в сети Интернет, происходит свободное общение педагога и обучающихся в социальных сетях, по электронной почте, посредством видеоконференции или в общем чате. Кроме того, дистанционное обучение позволяет проводить консультации обучающегося при самостоятельной работе дома. Налаженная система сетевого взаимодействия подростка и педагога, позволяет не ограничивать процесс обучения нахождением в учебной аудитории, обеспечить возможность непрерывного обучения в том числе, для часто болеющих детей или всех детей в период сезонных карантинов (например, по гриппу) и температурных ограничениях посещения занятий.

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;

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

Список литературы

Нормативные документы:

1. Федеральный закон «Об образовании в Российской Федерации» от 29.12.2012 N 273-ФЗ.

2. Стратегия развития воспитания в Российской Федерации на период до 2025 года. Распоряжение Правительства Российской Федерации от 29 мая 2015 г. № 996-р

3. Письмо Министерства образования и науки РФ от 18.11.2015г. № 093242. «О направлении Методических рекомендаций по проектированию дополнительных общеразвивающих программ (включая разноуровневые)»

4. Распоряжение правительства РФ от 04.09. 2014 № 1726-р «Об утверждении Концепции развития дополнительного образования детей»

5. «Основы законодательств РФ об охране здоровья граждан», утвержденные Верховным советом РФ от 22.07.1993 № 5487 - (ред. от 25.11.2009);

6. Федеральный закон от 24.07.1998 № 124-ФЗ «Об основных гарантиях прав ребёнка в РФ»;

7. Федеральный закон от «Об основах охраны здоровья граждан в Российской Федерации», 2011г.

8. Приказ Министерства просвещения России от 09.11.2018 г. № 196 «Об утверждении Порядка организации и осуществления образовательной деятельности по дополнительным общеобразовательным программам» (Приказ №1008 отменен).

9. Санитарно-эпидемиологические правила и нормативы СанПиН 2.4.4.3172-14.

Список литературы, использованной при написании программы:

1. Блох Джошуа. Java. Эффективное программирование. Effective Java. Programming Language Guide. изд. «Лори». 2014. – 310 с. 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. – 672 с. 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. – 496 с. 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 Developmente Edition. изд. Эксмо. 2011 г. 672 стр. ISBN 978-5-69950323-0.

5. Харди Брайн, Билл Филлипс.Программирование под Android. Android Programming: The Big Nerd Ranch Guide. изд. Питер. 2014. – 592 с. ISBN 978-5-496-00502-9, 978-0-321-80433-4.

6. Эльконин Д.Б. Детская психология: учеб. пособие для студ. высш. учеб. заведений / Д. Б. Эльконин; ред.-сост. Б. Д. Эльконин. — 4-е изд., стер. — М.: Издательский центр «Академия», 2007. — 384 с

Электронные ресурсы:

1. Портал обучения https://myitschool.ru

Приложение 1

Критерии оценки обучающихся

№ группы: _____ Дата: _____

№ п/п	ФИО обучающегося	Сложность продукта (по шкале от 0 до 5 баллов)	Соответствие продукта поставленной задаче (по шкале от 0 до 5 баллов)	Презентация продукта. Степень владения специальными терминами (по шкале от 0 до 5 баллов)	Степень увлеченности процессом и стремления к оригинальности (по шкале от 0 до 5 баллов)	Кол-во вопросов и затруднений (шт. за одно занятие)
1						
2						
3						
4						
5						
6						
7						
8						
9						
10						
11						
12						
13						
14						

Приложение 2

Бланк наблюдения за обучающимися

Группа_____ Педагог _____

№ п/п	ФИО	ПОКАЗАТЕЛИ					
		Внимателен в течение занятия	Использует базовую систему понятий	Проявляет инициативу, интерес в течение занятия	Идет на деловое сотрудничество	Аккуратно относится к материально-техническим ценностям	РЕЗУЛЬТАТ
1							
2							
3							
4							
5							
6							
7							
8							
9							
10							

За каждое согласие с утверждением 1 – балл

Приложение 3

План рассказа о проекте

1. Поприветствовать аудиторию. Представиться. Озвучить тему проекта.
2. Озвучить тему, актуальность, цели и задачи проекта.
3. Рассказать о выбранном наборе данных: источник, структура, размер.
4. Рассказать об использованных подходах, моделях и методах: причины выбора, структура, принцип работы.
5. Дать оценку качества работы модели по выбранным критериям.
6. Привести примеры работы модели.
7. В выводах озвучить, насколько достигнута поставленная цель и как усовершенствовать модель.
8. Поблагодарить за внимание.
9. Ответить на вопросы аудитории.

Приложение 4

Бланк оценки индивидуальных/групповых проектов

Проект является одним из видов самостоятельной работы, предусмотренной в ходе обучения по программе. Педагог оказывает консультационную помощь в выполнении проекта.

В комплект обязательных материалов, которые представляются обучающимся, входит: исходный код программы в архиве, презентация проекта.

Бланк оценки индивидуальных/групповых проектов

ФИО члена комиссии _____ Дата _____

Номер	Фамилия имя автора (-ов)	Название проекта	Время защиты	Критерий 1 **Актуальность проекта** 0-3 балла	Критерий 2 **Постановка проблемы** 0-3 балла	Критерий 3 **Целеполагание** 0-3 балла	Критерий 4 **Качество результата** 0-3 балла	Критерий 5 **Практическая реализация** 0-3 балла	Критерий 6 **Защита проекта** 0-3 балла	Итого

_____/_____

подпись расшифровка

Приложение 5

Бланк оценки индивидуальных проектов

Проект является одним из видов самостоятельной работы, предусмотренной в ходе обучения по программе. Педагог оказывает консультационную помощь в выполнении проекта.

В комплект обязательных материалов, которые предоставляются обучающимся, входит: исходный код в архиве, презентация проекта.

Бланк оценки индивидуальных/групповых проектов

ФИО члена комиссии _____ Дата _____

Номер	Фамилия имя автора (-ов)	Название проекта	Время защиты	Критерий 1 Актуальность проекта 0-3 балла	Критерий 2 Постановка проблемы 0-3 балла	Критерий 3 Целеполагание 0-3 балла	Критерий 4 Качество результата 0-3 балла	Критерий 5 Практическая реализация 0-3 балла	Критерий 6 Защита проекта 0-3 балла	Итого

_____/_____

подпись расшифровка

Приложение 6

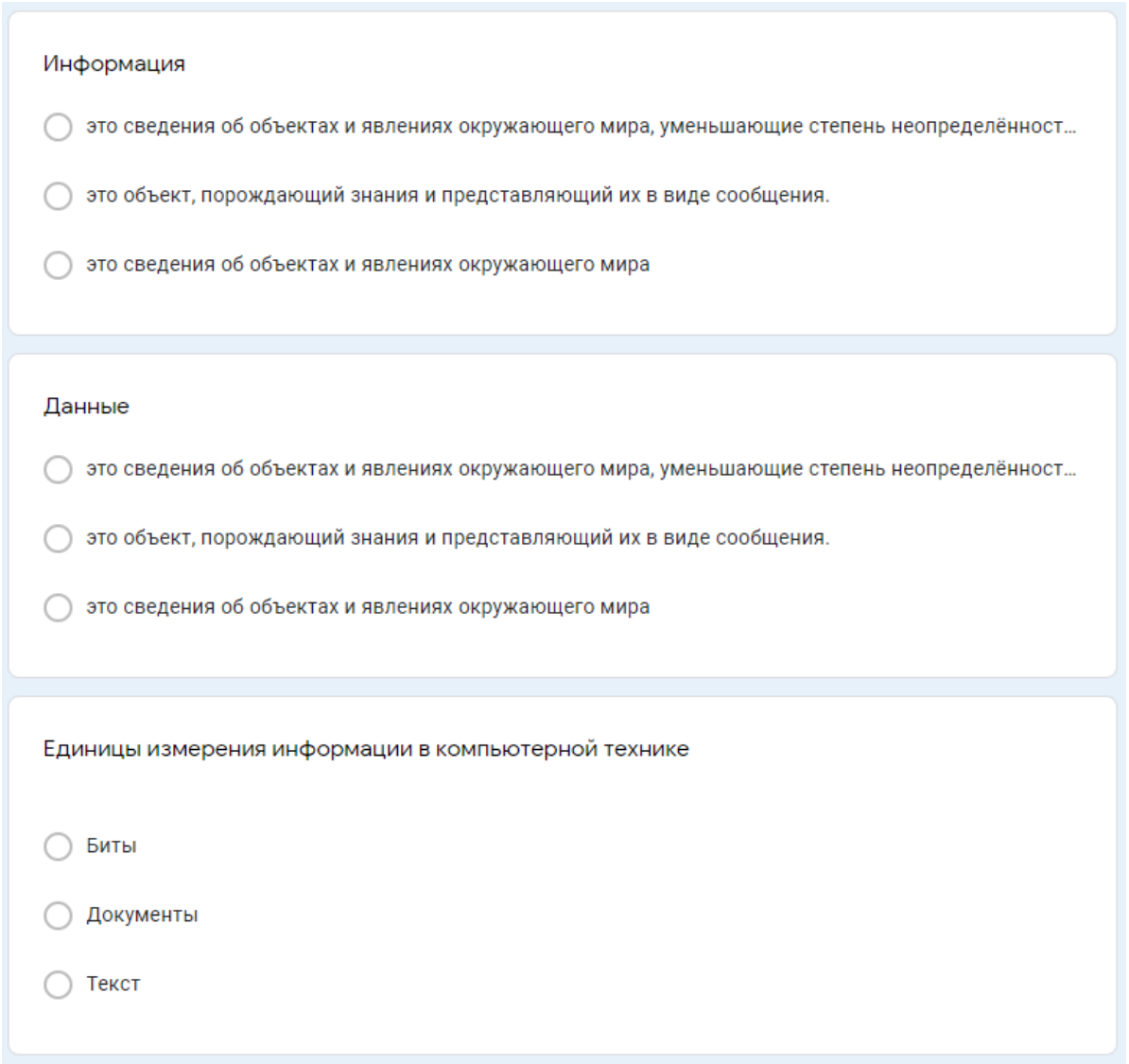
Мониторинг образовательной деятельности

Учебный курс предполагает следующие виды мониторинга образовательной деятельности:

- входной мониторинг
- мониторинг на занятиях
- промежуточный мониторинг
- итоговый мониторинг

Входной мониторинг

Предназначен для определения начального уровня подготовки ученика по настоящему курсу. Проводится в виде тестирования, возможна онлайн форма проведения тестирования.



Число - это

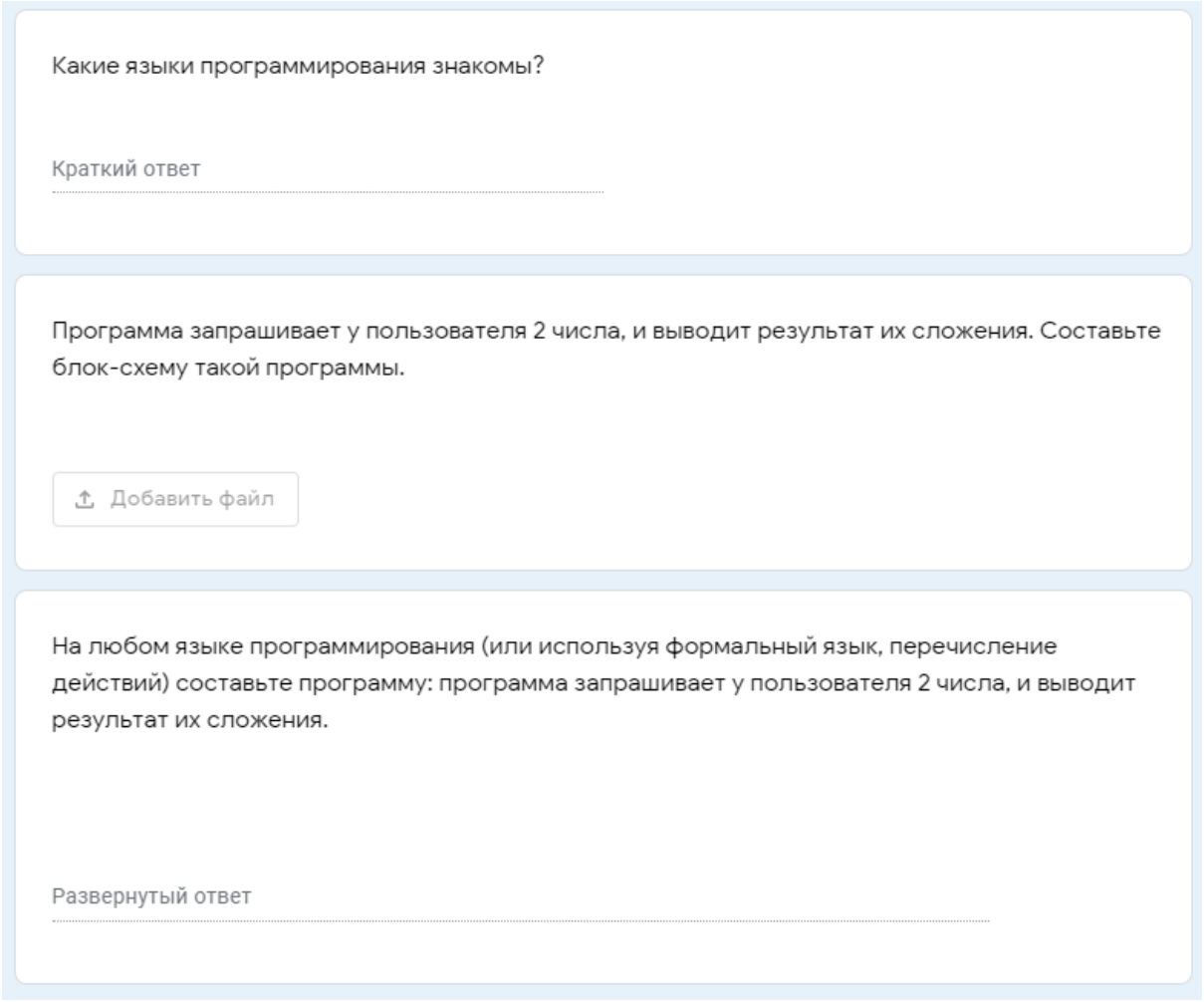
- ○ основное понятие математики, используемое для количественной характеристики, сравнения, нум...
- ○ знаки, используемые для записи количественного значения

Какие системы счисления знакомы: *

- ☐ Десятичная
- ☐ Двоичная
- ☐ Шестнадцатеричная
- ☐ Нет вариантов

Переменная -

- ○ именованная область памяти для хранения данных, которые могут изменяться в процессе исполн...
- ○ Изменяемые данные
- ○ Информация



Мониторинг на занятиях

Занятия предполагают интерактивный мониторинг усвоения теоретического материала и практических навыков. Мониторинг на занятиях осуществляется методом опроса и контроля применения полученных навыков по изучаемому курсу.

Методом наблюдения и общения на занятиях проводится мониторинг личностных характеристик учеников: коммуникабельность, ответственность, активность.

Промежуточный мониторинг

Проводится в завершении каждого модуля курса, предназначен для контроля усвоения материала. Промежуточный мониторинг проводится в виде тестирования. возможно проведение онлайн тестирования.

Тестирование по основам программирования:

Число - это *

Выберите правильные варианты

- [ ] основное понятие математики, используемое для количественной характеристики, сравнения, нум...
- [ ] знаки, используемые для записи количественного значения
- [ ] Нет верного

Какие утверждения относятся к десятичной системе счисления? *

- [ ] Система счисления с основанием 10
- [ ] Алфавит системы счисления состоит из {0,1,2,3,4,5,6,7,8,9,10}
- [ ] Алфавит системы счисления состоит из {0,1,2,3,4,5,6,7,8,9}
- [ ] Алфавит системы счисления состоит из {0,1}

Какие утверждения относятся к двоичной системе счисления? *

- [ ] Система счисления с основанием 10
- [ ] Система счисления с основанием 2
- [ ] Используется практически во всех современных компьютерах и прочих вычислительных электрон...
- [ ] Алфавит системы счисления состоит из {0,1}

Какое из утверждений верное? *

- [ ] int i; - объявлена целочисленная переменная
- [ ] double a = 3; - объявлена и проинициализирована переменная вещественного типа
- [ ] double a = 3,0; - объявлена и проинициализирована переменная вещественного типа
- [ ] char = f; - объявлена переменная символьного типа

Какое из утверждений верное? *

- [ ] String s=""; - объявлена переменная строкового типа
- [ ] String s= new String("Строка"); - объявлена переменная строкового типа
- [ ] string s="Строка"; - объявлена переменная строкового типа

Массив - это *

Выберите наиболее подходящий вариант определения

- ( ) структура данных, которая предназначена для хранения однотипных данных
- ( ) последовательность данных одного типа
- ( ) область памяти для хранения данных

Каким способом можно создать массив? *

- [ ] myArray = new int[10];
- [ ] String[] seasons = {"Winter", "Spring", "Summer", "Autumn"};
- [ ] String[] seasons = new String[] {"Winter", "Spring", "Summer", "Autumn"};

Каким способом можно получить длину строки, если str - переменная строкового типа? *

- ( ) str.length();
- ( ) str.Length();
- ( ) str.length;

Результатом работы программы будет *

```
public class Main {
    public static void main(String[] args) {
        int operation = 3;
        int rezultat=0;
        int x1=2, x2 = 5;
        switch (operation){
            case '1':
                rezultat = x1+x2;
                break;
            case '2':
                rezultat = x1-x2;
                break;
            case '3':
                rezultat = x1*x2;
                break;
            case '0':
                rezultat = x1/x2;
                break;
        }
        System.out.print(rezultat);
    }
}
```

- ◯ 7
- ◯ -3
- ◯ 10
- ◯ 0
- ◯ Другое...

Задайте двумерный массив 5X5 случайными числами в диапазоне [3,10]. Вывести на экран значения элементов массива в виде таблицы, после чего в отдельной строке вывести значениz сумм каждой строки массива. Чтобы получить число из диапазона [minRange;maxRange], можно использовать следующее выражение: (int) (minRange + Math.random() * (maxRange - minRange + 1)); *

Напишите код программы. Значения массива можно задать вручную, если сложность с генерацией случайных чисел

Развернутый ответ

Тестирование по основам ООП:

Наиболее подходящее определение понятия Объект *

- ○ Шаблон для создания данных в памяти компьютера
- ○ Экземпляр класса
- ○ Область памяти в которой хранится значение

Наиболее подходящее определение понятия Класс *

- ○ Нечто реальное, содержащее описание существующих сущностей
- ○ Шаблон или правила, по которому создаётся объект
- ○ Область памяти компьютера в торой хранятся структурированные данные

Поле класса: *

Выберите наиболее подходящее определение

- ○ Данные
- ○ Переменные определённого типа
- ○ Данные этого класса
- ○ Действия, производимые данным классом

Метод класса:

- ○ Данные
- ○ Переменные определённого типа
- ○ Данные этого класса
- ○ Действия, производимые данным классом

Инкапсуляция *

- описание правила построения объекта
- принцип объектно-ориентированного программирования, в рамках которого возможно описание н...
- требование, что объект должен содержать в себе всё необходимое для своей жизни, но не более т...
- способность предоставлять один и тот же интерфейс для различных типов данных

Наследование *

- принцип объектно-ориентированного программирования, в рамках которого возможно описание н...
- требование, что объект должен содержать в себе всё необходимое для своей жизни, но не более т...
- способность предоставлять один и тот же интерфейс для различных типов данных
- особый класс, описывающий методы без их реализации

Полиморфизм *

- принцип объектно-ориентированного программирования, в рамках которого возможно описание н...
- требование, что объект должен содержать в себе всё необходимое для своей жизни, но не более т...
- способность предоставлять один и тот же интерфейс для различных типов данных
- особый класс, описывающий методы без их реализации

Интерфейс в ООП *

- принцип объектно-ориентированного программирования, в рамках которого возможно описание н...
- требование, что объект должен содержать в себе всё необходимое для своей жизни, но не более т...
- способность предоставлять один и тот же интерфейс для различных типов данных
- особый класс, описывающий методы без их реализации

В соответствии с принципом инкапсуляции *

- ○ Можно обратиться напрямую к полям объекта
- ○ К полям следует обращаться через соответствующие методы

Модификатор доступа public *

- ○ Открывает доступ (делает видимым) к членам класса из любого места программы
- ○ Помечает член класса как недоступный из вне
- ○ Указывает ссылку на сам объект

Модификатор доступа private *

- ○ Открывает доступ (делает видимым) к членам класса из любого места программы
- ○ Помечает член класса как недоступный из вне
- ○ Указывает ссылку на сам объект

Ключевое слово static *

- ☐ Является ссылкой на экземпляр класса
- ☐ Делает член класса общим для всех экземпляров класса
- ☐ Открывает доступ (делает видимым) к членам класса из любого места программы
- ☐ Является ссылкой на родительский класс

Ключевое слово this *

- [ ] Является ссылкой на экземпляр класса
- [ ] Делает член класса общим для всех экземпляров класса
- [ ] Открывает доступ (делает видимым) к членам класса из любого места программы
- [ ] Является ссылкой на родительский класс

Ключевое слово super *

- [ ] Является ссылкой на экземпляр класса
- [ ] Делает член класса общим для всех экземпляров класса
- [ ] Открывает доступ (делает видимым) к членам класса из любого места программы
- [ ] Является ссылкой на родительский класс

Отметить верные утверждения о конструкторах класса *

- [ ] Начинается с заглавной буквы и совпадает с именем класса, в котором описан
- [ ] Может иметь параметры
- [ ] Может не иметь параметров
- [ ] Всегда должен возвращать результат

Возможно ли описать в классе 2 метода с одинаковым именем? *

- ( ) Да
- ( ) Нет
- ( ) Да, если параметры таких методов будут различаться

Ключевое слово extends *

- описывает наследование от класса
- описывает наследование от интерфейса

Ключевое слово implements *

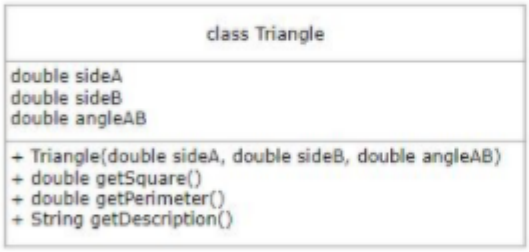
- описывает наследование от класса
- описывает наследование от интерфейса

Возможно ли наследование от нескольких классов? *

- Да
- Нет

Опишите на языке Java представленный класс *

Реализацию методов можно опустить, т.е. оставить пустые {}



Развернутый ответ

Итоговый мониторинг

Работа над персональным проектом — предполагает построение цели, задач для ее достижения, и выполнение по персональному ТЗ.

Аннотация

Задача инновационного развития программного обеспечения требует соответствующей образовательной среды, в том числе создания оптимальных условий детского технического творчества. Одной из наиболее инновационных областей в сфере детского технического творчества является мобильная разработка.

Для дальнейшего развития мобильных приложений существует широкий выбор направлений разработки. Каждому ребёнку интересно, как устроена Java платформа, как работает Java приложение на любой платформе и на смартфоне в том числе.

Изучение языка программирования Java по данной программе обучения даёт возможность пользователю освоить базовые навыки использования языка программирования, понять его особенности использования и выполнения на различных платформах.

Разработка мобильных приложений на базе Android на сегодняшний день. очень востребована ввиду высокой популярности данной ОС. Поэтому обучение по данной программе – это самый первый, но важный шаг в изучении основ программирования на языке Java, для создания проектов и простейших программ в среде разработки на его основе.

Программа рассчитана на обучающихся 14–17 лет.