Ключевые моменты по урокам курса Основы Java

### Урок 1

1. Что такое компьютерная программа.

Компьютерная программа – это последовательность команд, на понятном компьютеру языке, которые будут выполнены компьютером при запуске такой программы.

1. Зачем нужен язык программирования.

Язык программирования необходим для того что бы человек мог свои мысли переводить на машинный язык, понятный компьютеру, реализуя тем самым своего рода диалог между человеком и машиной.

1. Что такое переменная.

Переменная – это именованная область памяти компьютера в которой можно хранить данные определенного типа, получать к ним доступ и изменять их в ходе выполнения компьютерной программы, где она используется.

1. Простые типы в Java.

К просты типам в Java обносяться: int, long, double, char, boolean

1. Что такое строка в Java.

Строка в Java это экземпляр объектного типа String. Записывается как любая последовательность символов заключенная в двойные кавычки: “Abc123!%,.:?”.

### Урок 2

1. Что такое метод, зачем он нужен.

Метод это именованный блок кода, который реализует определенную логику. Метод можно объявить, описать его тело и вызвать, когда это необходимо, из другого участка кода. В зависимости от входящих данных - передаваемых в метод параметров, мы будем получать разные результаты, используя один и тот же блок кода - метод. Использование методов реализует идею повторного использования кода. То есть написанный один раз код метода, можно использовать в дальнейшем, где и сколько угодно раз, просто помня, что возвращает или выполнят, используемый метод, получив на вход те или иные параметры. При этом необязательно знать или помнить саму реализацию метода.

Синтаксис:

<return type> | void <name>(<parameter1>,... <parameterN>) {

//Body

[return <result>;]

}

1. Что такое ссылка.

Ссылка это переменная, которая хранит адрес другой переменной, в которой, в свою очередь хранятся данные.

1. Как осуществляется передача параметров в метод (простые и ссылочные типы).

Передача параметров в метод осуществляется по значению. При передаче переменных простых типов данных в метод в качестве параметров, значения этих переменных просто копируются в вызываемый метод, при этом передаваемые параметры останутся неизменными, в той части программы (напр. в main()), откуда вызывается метод.

При передаче параметров объектных или ссылочных типов в метод, ситуация аналогичная, только при этом передаются копии значений ссылок на объекты. И если внутри метода мы переопределим переданную ссылку, то внешняя ссылка не изменится, а поменяется только значение её копии внутри метода.

1. Зачем нужен оператор if.

Условный оператор if реализует логические ветвления и позволяет Java программе, в зависимости от условия, выполнить то или иное действие, основываясь на значении (true/false) булевой переменной или логического выражения.

1. Зачем нужны циклы.

Жизнь циклична. Часто бывает необходимость неоднократно выполнять те или иный действия в программе. Для этого существую циклы. В Java мы научились использовать while и for.

### Урок 3

1. Что такое массив и как он представлен в памяти компьютера.

Массив это последовательность проиндексированных элементов одного типа. Доступ к каждому элементу массива осуществляется по его индексу. Массив имеет заданную длину или размер, которая соответствует количеству всех его элементов. Индекс первого элемента массива равен 0, последнего – на единицу меньше размера массива.

Массив представлен в памяти как переменная – имя массива, которая содержит адрес первого элемента массива с индексом [0], за которым далее последовательно распложены остальные элементы массива. Вся эта цепочка занимает определенный объём памяти, который зависит от типа элементов массива, то есть числа байт, которое отводиться под каждый элемент массива и общего числа элементов массива.

1. Что из себя представляет многомерный массив.

Многомерный массив это массив, элементы которого являются массивами. Чаще всего используется двумерные массив. Такой массив можно представить в виде матрицы, где доступ к элементам осуществляется по двум индексам – номеру строки и номеру столбца ячейки в которой находиться элемент массива. Правда это условное представление. На самом деле в памяти компьютера, например в случае с двумерным массивом, мы имеем два массива. Каждый элемент первого, назовем его основным, массива хранит адрес первого элемента массива, который храниться в этом элементе. Если у нас трех-, четырех- и т.д. мерный массив данный принцип организации многомерных массивов в памяти компьютера просто раскручивается по цепочке.

### Урок 4

1. Что такое компиляция.

Компиляция – это трансляция кода программы, на языке высокого уровня в машинный код, понятный компьютеру, которую выполняет компилятор. Исходный Java-код компилируется в байт-код исполняемый Java Virtual Machine (JVM).

1. Запуск программ из командной строки, зачем это нужно.

Интерфейс командной строки (CLI) менее требовательный к ресурсам компьютера, чем графический (GUI).

Иногда графический интерфейс просто не нужен, например, при работе с сервером.

Командная строка обеспечивает более быстрый доступ к некоторым возможностям ОС, а иногда — это единственный способ запустить тот или иной процесс.

1. Что такое класс и объект.

Класс - это способ организации полей (атрибутов) и методов, инкапсулированных (упакованных в единый компонент) внутри структуры класса со своим пространством имен, позволяющий описать структуру и свойства какого-либо объекта из реальной жизни.

Объект – это экземпляр класса в памяти компьютера, обладающий определённым состоянием (значением полей) и поведением (методами).

1. Зачем нужны конструкторы.

Конструкторы необходимы для создания объектов (экземпляров) класса в памяти компьютера. Не всегда удобно инициализировать все переменные класса при создании его экземпляра. Иногда проще, чтобы какие-то значения были бы созданы по умолчанию при создании объекта. По сути конструктор нужен для автоматической инициализации переменных. Даже если мы явно не указываем конструктор в классе, то Java автоматически создаст его для использования по умолчанию.

1. Зачем скрывать переменные класса (делать их private) и обращаться к ним через getter, setter.

Сокрытие переменных класса с использованием модификатора доступа private используется для обеспечения целостности данных. Например, чтобы разработчик класса наследника не мог напрямую и без надобности изменять значения переменных/полей родительского класса.

1. Что такое enum.

Enum – это перечисление – специальный тип данных используемый для ограничения множества допустимых значения. Например, день недели может иметь 7 разных значений, месяцев в году - 12, а времен года – 4.

### Урок 5

1. Что такое наследование.

Наследование — механизм, позволяющий описать новый производный класс на основе уже существующего базового (родительского) класса. Класс потомок может дополнить или расширить возможности класса родителя, а также может использовать родительские методы и свойства.

Для того что бы определить иерархию классов, необходимо взять все классы одного типа и, выделив в них общие характеристики и свойства, вынести их в отдельный класс. Главный наследуемый класс в Java называют суперклассом (super). В Java у каждого класса должен быть только один класс родитель, а вот классов потомком может быть сколько угодно. При этом доступна любая вложенность.

1. Как работает method overriding.

Method overriding позволяет в классе потомке написать специфическую реализацию метода, который уже реализован в родительском классе. При переопределении метода из родительского класса в классе потомке, описывается метод с точно таким же названием, типом возвращаемого результата и принимаемыми параметрами. Вызов метода происходит по иерархической цепочке снизу вверх. Например, если мы вызываем метод, который содержится в классе родителе и, который переопределен в классе потомке, то вызывается именно реализация метода из класса потомка.

1. Отношение Наследник — Родитель.

Когда мы наследуем один объект от другого, это означает, что объект наследник сам по себе является объектом родителем (parent). Когда мы создаем экземпляр конкретного класса наследника, то экземпляр наследник имеет свойства и функции родителя, которые в него встраиваются, что позволяет обращаться к нему как к родителю, при этом он имеет такое же поведение. Например: Flute is an (extends) Instrument означает, что экземпляр класса Fluent будет иметь весь такой же функционал и такое же поведение, что и экземпляр класса Instrument.

При наследовании используется механизм upcasting – преобразование типа вверх по иерархии наследования. Upcasting позволяет передать в метод объект класса, который наследует класс, принимаемый этим методом.

1. Delegation (передача полномочий), Composition (включение).

Delegation (делегирование) – это передача полномочий выполнения методов одного класса – другому классу. При делегировании говорят о втором типе наследовании “has a”, когда один объект включает в себя другой объект. При организации делегирования необходимым условием есть реализация возможности обратиться ко всем не private методам класса, которому делегируются полномочия. В противном случае, если нет обращения ко всем не private методам класса, который включен в реализуемый класс, говорят о включении (composition) одного класса в другой. Иными словами делегация (delegation) – это частный случай включения (composition), при котором в классе, который делегирует управление, реализуются все не private методы класса, которому делегируется управление, путем написания аналогичных по названию и параметрам методов, которые просто вызывают методы класса которому делегируется управление.

Наиболее популярные случаи использования delegation: когда удобней построить иерархию классов путем включения одного класса в другой, избегая полного расширения одного класса другим; когда необходимо добавить в класс дополнительный функционал, не изменяя при этом сам класс, например, когда нет доступа на изменение, метод/класс является final или такова техническая реализация.

1. Статические данные, ключевое слово static.

Ключевое слово статик может применяться к переменным, методам и классам.

Статические переменные и методы — это переменные и методы которые принадлежат самому классу, а не его объекту/экземпляру и создаются, когда JVM загружает сам класс, при первом обращении к нему или его упоминании в программе. Статические переменные и методы можно вызывать не создавая экземпляр класса. При реализации статических методов нельзя использовать не статические переменные и методы класса.

Чаще всего статический переменные используются, когда необходимо реализовать константы, при этом после static добавляется ключевое слово final. Статические методы чаще всего применяться при реализации классов утилит, для обработки объектов этого же класса (пример класс String), при реализации метода по созданию объектом самого себя.

1. Зачем нужны классы-обвёртки (wrappers).

Классы-обвёртки (wrappers) необходимы для организации работы с простыми/примитивными типами, как с полноценными объектами, расширяя их функционал. Wrappers: Boolean, Integer, Long, Double, Char, Byte, Short, Float.

### Урок 6

1. Как порождаются исключительные ситуации и как на них реагировать.

Исключительные ситуации могут возникать независимо от программиста (не хватило оперативной памяти для выполнения программы или места на жестком диске) – это неконтролируемые исключения – класс Error, которые устраняются путем настройки JVM или самой ОС, или же путем установки нового железа. Известные программисту ситуации, которые могут возникать в программе – это контролируемые исключения – класс Exception (все исключительные ситуации, которые обязательно необходимо обрабатывать на этапе компиляции) и RuntimeException (extends Exception, исключительные ситуации, которые необязательно обрабатывать), которыми можно управлять посредством конструкций throws Exception {} , try {} catch (Exception e) {} | finally {}

1. Что такое полиморфизм.

Полиморфизм – это способность объектов с одинаковой спецификацией иметь различную реализацию. Также это способность объекта использовать методы производного класса, который не существует на момент создания базового. Ярким примером полиморфизма является использование классов-наследников, как классов-родителей. При этом, если в классе-наследнике был переопределен какой-то метод, то вызовется именно он. Концепцией полиморфизма является идея «один интерфейс, множество методов».

1. Что такое абстрактный класс и зачем он нужен.

Абстрактный класс (abstract class) – это специальный класс экземпляр (instance) которого мы не можем создать. Абстрактные классы используются для хранения общих методом и характеристик (переменных) иерархии классов с целью избегания дублирования кода. Часто абстрактный класс используют для инициализации базового состояния объектов класса.

1. Что такое интерфейс и зачем он нужен.

Интерфейс – это синтаксический механизм Java позволяющий определить поведение объекта.

Свойства интерфейса (поля/переменные) должны быть константными и статическими (final static), а методы общедоступными (public). Чаще всего в интерфейсе описывают лишь сигнатуры методов. Интерфейс в Java заменяет множественное наследование, которого здесь нет, соответственно, конкретный класс может наследовать (extends) только один абстрактный класс. Зато имплементировать (implements) разных интерфейсов этот класс может, сколько угодно.

State Pattern – это один из шаблонов проектирования, который используется когда: поведение объекта зависит от его текущего состояния и может меняться в процессе выполнения; производятся сложные вычисления которые зависят от состояния объекта. Это делает переход в другое состояние явным и сохраняет в одном месте описание поведения объекта в конкретном состоянии.

### Урок 7

1. Зачем нужны коллекции.

Коллекции или контейнеры — это классы позволяющие хранить и производить операции над совокупностью объектов. Коллекции используются для хранения и манипуляции данными, обеспечивая агрегацию одних объектов другими.

1. Зачем нужны методы equals и hashCode.

Метод equals – это метод класса Object, который сравнивает объекты по значению.

equals(Object):boolean

1) Reflexive: x.equals(x) == true

2) Symmetric: x.equals(y) == true => y.equals(x) == true

3) Transitive: x.equals(y) == true and x.equals(z) == true => y.equals(z) == true

4) Consistent: x.equals(y) == true – during repeated call

5) x.equals(null) == false

6) x = new Object(); y = x; x.equals(y) == true

1. Что из себя представляет список.

Список (List) – это коллекция для работы с объектами схожа с массивом, но имеющая ряд отличий. Характеристики списка: доступ по индексу; сохраняется очередность объектов (куда положили, там объект и храниться); не нужно указывать кол-во объектов, которые будут храниться списке; список хранит любые объекты, кроме примитивных типов; список может хранить дубликаты и null.

Список хранит объекты в специальной структуре данных, которая называется Node. В Node храниться сам объект (object) и ссылка (reference) на следующий Node.

1. Что такое Iterator.

Итератор (Iterator) – это паттерн позволяющий получить доступ к элементам любой коллекции без вникания в суть ее реализации. Iterator реализует методы: next() – возвращает следующий элемент списка; hasNext() – возвращает true, если в списке есть еще один элемент и false – если нет; remove() – удаляет элемент списка.

1. Как сравнить элементы коллекции.

Для сравнения элементов коллекции используется специальный интерфейс Comparator. Comparator содержит метод int compare(Object o1, Object o2), который возвращает -1 – если о1 < o2, 0 – если о1 == о2, 1 – если о1 > о2.

1. Основные коллекции, где какую лучше использовать

В библиотеке коллекций Java (Java Collections Framework) существуют такие базовые коллекции (интерфейсы):

Iterable – позволяет осуществлять перемещение по коллекции, посредством итератора (Iterator).

Collection (extends Iterable) – содержит основной набор методов для манипуляции с данными, такие как вставка (add, addAll), удаление (remove, removeAll, clear), поиск (contains). Collection наследуют (extends) List и Set.

List (список) – это коллекция, содержащая отсортированный набор объектов, которые могут дублироваться. В List сохраняется порядок объектов – куда положили, оттуда взяли.

Set (множество) – это коллекция, которая не содержит дубликатов, не гарантирует очередность объектов, как правило, не содержит null. Стандартные имплементации: HashSet – обеспечивает быстрый поиск; TreeSet – обеспечивает постоянную очередность элементов, благодаря древовидной структуре данных.

Map (карта) – это коллекция, которая содержит объекты в виде пар "ключ — значение", некий аналог ассоциативного массива (associative array), когда значение можно получить по определённому ключу. Map, в отличии от List и Set, не реализовывает Iterable интерфейс, то есть не позволяет осуществлять доступ к элементам посредством итератора. Map можно рассматривать в трех представлениях: коллекция ключей (Keys Set); коллекция значений (Values); коллекция пар ключ-значение (Key-Value). Все объекты в Map коллекции содержаться в специальной структуре данных, которая называется Entry (содержит key, value, hash).

Для того что бы определиться с выбором конкретной коллекции, необходимо определить, как часто мы будем использовать конкретную операцию и как быстро она должна выполняться.

Если нам необходим доступ по индексу, то логично выбрать Array (для простых типов) и ArrayList (для объектов). Array или ArrayList следует использовать, когда вставки элементов в коллекцию планируется выполнять не часто, и в тоже время, планируется много операций поиска и выборки по индексу, а также замен элементов коллекции (например сортировка элементов). Когда же необходимо часто делать вставки элементов, а доступ по индексу необходим гораздо реже, то следует использовать LinkedList.

Если в коллекции будут содержаться уникальные объекты, то это Set. В случае, когда нам необходим быстрый поиск, нужно использовать HashSet. Если же нам необходимо хранить уникальные объекты в отсортированном в какой-то способ виде, следует использовать TreeSet.

Если нам необходим доступ по ключу, то используется Map. HashMap (имплементация на основе hash-таблиц с использованием bucket-ов – “ведер”) используется для быстрого поиска, а TreeMap (имплементация на основе дерева) используется когда необходима сортировка ключей определенным образом.

### Урок 8

1. Как правильно работать с датой и временем.

Для работы с датой и временем в Java существуют классы java.util.Date и java.util.Calendar.

Класс Date хранит время в миллисекундах (число long) начиная с 01.01.1970 и имеет конструктор по умолчанию, который возвращает текущее время согласно настройкам компьютера. Для форматирования вывода даты и времени используется класс java.text.SimpleDateFormat. При создании шаблона для отображения даты dd – означает день, MM – месяц, yyyy – год, hh – часы и mm – минуты, ss – секунды.

Для работы с датой в рамках календаря используется абстрактный класс java.util.Calendar. Конструктор по умолчанию GregorianCalendar() возвращает календарь для текущего дня согласно настройкам компьютера. Класс Calendar позволяет получать названия месяцев и дней недели, увеличивать или уменьшать различные параметры текущей даты, а также получать их.

1. Основные идеи в построении GUI.

Основными средствами создания GUI в Java являются библиотеки AWT и Swing.

Библиотека AWT (Abstract Window Toolkit) содержит средства необходимые для работы с геометрией, текстом, картинками и вывода на печать. Основными классами AWT являються классы Graphics и Graphics2D.

Библиотека Swing содержит стандартные компоненты графического интерфейса (окна, поля ввода, выпадающие списки, таблицы и т.д.). Также, библиотека Swing содержит средства для обработки событий (ввод с клавиатуры, клин мыши, прокрутка колесика и т.д.).

Основные моменты при построении GUI:

1. При построении GUI создается контейнер JFrame, который состоит из двух областей Menu Bar (верхняя часть любого окна программы) и Content Pane, контейнера, которой и принимает все остальные компоненты GUI.
2. Все объекты библиотеки Swing, которые служат для создания графического интерфейса, наследуются от класса JComponent. Они делятся на два типа – контейнеры (JPanel, Content Pane), которые визуально не видны, но служат для группировки остальных компонент, и визуальные компоненты GUI (JLabel, JTextField, JComboBox, JButton и т.д.).
3. Для того что бы добавить какой-то визуальный компонент GUI, необходимо создать объект этого класса и добавить его в Content Pane. После чего этот компонент отобразиться внутри окна GUI.
4. Для группировки компонент внутри Content Pane используется объект JPanel – стандартный контейнер, куда помещаться необходимые визуальных компоненты GUI. JPanel, в свою очередь, помещается в Content Pane.

### Урок 9

1. Зачем нужен Adapter.

Adapter (адаптер) адаптирует поведение одного объекта под нужны другого объекта. Существует два распространенных варианта реализации адаптера: наследования класса (adapter by class) и включение объекта (adapter by object). Adapter by object хороший вариант для адаптации старой библиотеки или ранее реализованной логики под новый интерфейс. Adapter by class применим только в случае, когда необходимо адаптировать класс, но не интерфейс.

Хорошим примером применения Adapter Pattern является адаптирование интерфейса путем использования абстрактного класса адаптера. Например у нас есть интерфейс с несколькими методами, но нам нужен только один. В данном случае мы создаем абстрактный класс адаптер (abstract class OurAdapter) и переопределяем (override) в нем все методы нашего интерфейса с пустым телом, если какой-то метод что-то возвращает, то пишем return 0 или null, в зависимости от типа возвращаемого значения. Далее, когда нам нужно использовать наш интерфейс в адаптированном виде, мы используем не сам интерфейс, а его адаптер, переопределяя необходимый нам метод(ы). В Java таки образом реализован MouseAdapter.class в связке с MouseListener.class.

1. В чём суть модели Publish / Subscribe

В моделе Publish / Subscribe:

- Publisher ничего не знает о своих Subscriber(ах);

- Subscriber знает все о своих Publisher(ах);

- Subscriber на прямую не отслеживает события Publisher(а), за него это делает «третье лицо»;

- Publisher на прямую не уведомляет Subscriber(а) о своих события, за него это делает «третье лицо»;

- Subscriber реагирует на события Publisher(а).

Работу модели Publish / Subscribe обеспечивает Observer Pattern, позволяя объектам реагировать на изменение состояния других объектов. Observer Pattern исключает прямую зависимость объекта наблюдения и наблюдателя.

Наиболее распространённые сценарии использования Observer Pattern:

Изменения в одном объекте влечет за собой изменения в других объектах, причем их количество заведомо не известно. К примеру, автомобиль, итоговая стоимость которого состоит из стоимости комплектующих и, если стоимость какой-то комплектующей поменялась, например, стоимость парковочного радара, то стоимость всех автомобилей, на которых он установлен, автоматически пересчитывается.

Объект должен оповестить другие объекты без каких-либо сведений об их структуре. Например, подписка на новые публикации на сайте для зарегистрированных пользователей, которая позволяет при каждой новой публикации, отправлять уведомление на e-mail подписчика, о котором с технической точки зрения ничего не известно.

1. Реакция на события, принцип работы.
2. В чём суть MVC.

Model View Controller (MVC) – шаблон (pattern) проектирования, который разделяет программу или какую-то ее часть (модуль) на три логические части: Model – отвечает за данные; View – отвечает за отображение/представление данных; Controller – реагирует на действия View (например клик мышкой) и запускает механизм, отвечающий за изменение Model (чаще всего это обращение к бизнес логике). Model, View и Controller взаимодействуют между собой определенным образом.

Например в окне JFrame происходит событие – клик мышкой, при этом запускается action, который отлавливает Listener и производит вызов (dispatch) Business Logic, которая, в свою очередь, каким-то образом меняет (update) данные в Model, которые в итоге отображаются в окне JFrame. В данном примере JFrame отвечает за логическую часть View, Listener является контроллером, а данные – Model.

MVC Pattern предоставляет два основных преимущества:

1. Данные отделяются от их представления – Model не зависит от View.
2. Данные можно отобразить несколькими способами. Например, таблица и диаграмма. Также, к примеру, можно создать несколько GUI одной программы и использовать их в зависимости от ОС или же от уровня доступа.

MVC Pattern используется повсеместно, где необходимо построение GUI, а также стандартно при создании веб-приложений.