Инициализация и завершение

- Основные проблемы с безопасность при инициализации и завершении
- Инициализация с помощью реализации метода initialize() или при создании конструктора
- Конструктор по умолчанию - конструктор без аргументов, если есть с арг, то должен быть объявлен и по умолчанию
- У конструктора отсутствует возвращаемое значение
- Оператор new возвращает ссылку на вновь созданный объект

- Перегрузка методов - overloading - использование одинаковых имен с разными (уникальными!!) аргументами
- При перегрузке примитивами - если входящее значение меньше требуемого, оно может автоматически расширятся
- Если входящее значени больше требуемого - компилятор выдаст сообщение об ошибке
- Перегрузка по возвращаемым значениям не возможна

- Ключевое слово this
- Используется внутри не-статического метода, предоставляет ссылку на объект, для которого был создан метод
- Употребляется в случае, когда вам надо явно солаться на объект
- С помощью слово this можно осуществить вызов конструктора из конструкторов

- Слово static
- Нельзя вызвать нестатические методы из статических (а наоборот можно)

- Финализация и отчистка мусора
- Уборщик знает когда убирать с использованием new, но как без него?
- Без него - метод finalize() используемый в классе

- Объекты могут быть не переданы уборщику мусора
- Уборка мусора не есть уничтожение
- Уборка мусора - относится только к памяти

- Finalize была введена в язык, чтобы сделать возможным создание с необычными механизмами выделения памяти (вызов не джава кода из программы джава)
- Применение метода finalize связано с условием готовности объекта - в той точке, где объект готов к удалению, его удаление должно быть безопасно

- Работа уборщика мусора
- Уборка мусора приводит к ускорению создания объектов
- Уборщик мусора смещает объекты к началу кучи -> позволяет избежать фрагментацию помяти
- Адаптивный механизм уборки мусора - JVM обращается с используемыми объектами согласно опр варианту действий, например - остановит-и-копировать - работа приостанавливается все живые объекты копируются из одной кучи в другую
- Копирующие уборщики медленные тк - предпологают наличие двух куч + постоянное копировние. При работе с блоками используется счетчик поколениц, следящий за использованием блока. Подходит для создание множества временных объектов с коротким сроком жизни
- Уборщики пометить и убрать - обнаружение всех ссылкок на живые объекты, мертвые удаляется - подходит для долгоживущих объектов
- Переключение между уборщиками есть адаптивный метод

- Инициализация членов класса
- В случае локального использования примитивов отстутсвие инициализации замечается и наказывается
- Если примитив поле класса, то без инициализации - значение по умолчанию
- ЯВНАЯ ИНИЦИАЛИЗАЦИЯ
- В случае инициализации объектов
- При отсутствие таковой и использовании - исключение
- Инициализация через new, через функции
- Инициализация конструктором - придает большую гибкость, так как появляется возможность
- инициализации во время работы программы. Но можно инициализировать и до конструктора
- Порядок инициализации определятся очередностью следования элементов, но по итогу то,
- что в конструкторе

- Инициализация статических данных
- Данные статических полей - в единственном экземпляре
- Значение по умолчанию для ссылок - null
- Если инициализация в конструкторе, то статич. иниц только при необходимости
- Сначала иниц статические, потом не статические

- Явная инициализация статических членов
- статический блок - static{} - группировка несколких статич иниц
- Статич иниц выполняются при обращении к статическому объекту
- Выполняется при обращении к одному из блоков

- Нестатич иниц
- {} - блок инициализации, гарантирует, что иниц будет выполнена вне зависимоти от конструктора
- Секция инициализации выполняется раньше любых конструкторов

- Иниц массивов
- Определяются с помощью оператора индексирования []
- Чтобы определить ссылку на массив - указать вслед за типом []
- Массив без инициализации используется для
- присвоений одному массиву значений другого
- Поле длинна - неизменяемое
- При создании массива непримитивных типов - создается массив ссылок, пока в каждой не будет

- Списки аргументов переменной длины
- для вывода - print(Object... args), где args - массив + foreach
- избавляют от необходимости явной записи синтаксиса массивов
- При отстутсвии аргументрв - массив нулевой длинны
- При использовании переменной длинны и примитивов может быть ошибка в перегрузке, тк
- не понятно что именно использовать (какой массив)

- Перечисления enum
- Ключевое слово enum - упрощает работу при группировке и использовании перечеслимых типов
- Раньше - набор целочисленных констант
- чтобы создать перечисление - создать ссылку на перечислемый тип и присвоить ее перменной
- Удобно использовать в switch
- Перечисление как дополнительный способ создания типа данных и работы с полученными результатами

- *Финализация для графических и файловых дескрипторов
- *Конструкторы позволяют осуществлять композицию и наследование