**JAVA FUNDAMENTALS**

JAVA – вычислительная платформа и объектно-ориентированный, платформенно – независимый ЯП.

JAVA language specification – тех. описание семантики и синтаксиса

JAVA API – предопределенные классы и интерфейсы .http://docs.oracle.com/javase/7/docs/api/

Редакции: JAVA SE, JAVA EE, JAVA ME

JVM – интерпретирует байт-код, описываемый в класс-файлах. (bin)

JRE – среда выполнения, для запуска приложений.

JDK – набор утилит, стандартные библиотеки. (java, javac,jar,javadoc … )

**Packages**

Пакеты – контейнеры классов, для разделения пространства имен. Физически – папка на диске.

package имя\_пакета.имя\_подпакета;

При компиляции, поиск пакетов осуществляется:

- в рабочем каталоге;

- параметр переменный classpath;

- через параметр компилятора –classpath;

Правила именования:

1) с маленькой буквы

2) если 2 слова – отдельный подпакет или через подчеркивание.

3) by.epam.project\_name.package\_name (4-8 вложений)

Import – для подключения пакета.

import имя\_пакета.имя\_подпакета.\*;

import имя\_пакета.имя\_подпакета.имя\_класса;

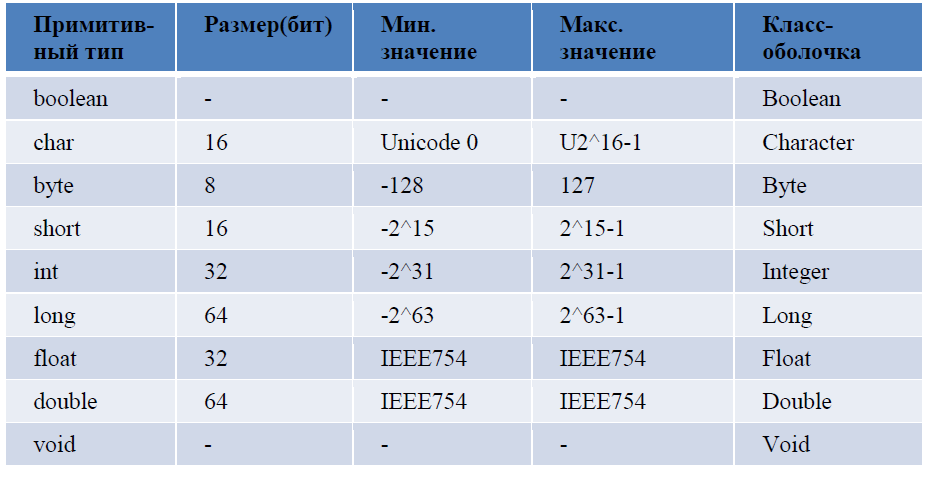
*Лучше импортировать сущности. Если указать \* - выбор сущностей только из данной директории, подпакеты и надпакеты не выбираются.*

**Модификаторы доступа**

Private > Friendly > Protected > Public

**Типы данных**

**Примитивные типы**



Зн по умолчанию (false, ‘\u0000’(null),(byte) 0 , (short) 0, 0 , 0L, 0.0f , 0.0d, -)

Особенности:

- размер одинаков для всех платформ.

- размер boolean неопределен

- невозможно преобразование между Boolean и другими типами.

* *В именах переменных не используются арифметические, логические операторы и #.*
* *С осторожностью использовать операторы ++ и --, т.к. возможно присвоение значения более длинного типа.*

**Преобразование примитивных типов**

Явное (сужающее) - результирующий тип – меньший диапазон значений, чем исходный тип.

Неявное(расширяющее) - результирующий тип – больший диапазон значений. (автоматически)  
 возможна потеря данных при:int>float long>double int>double

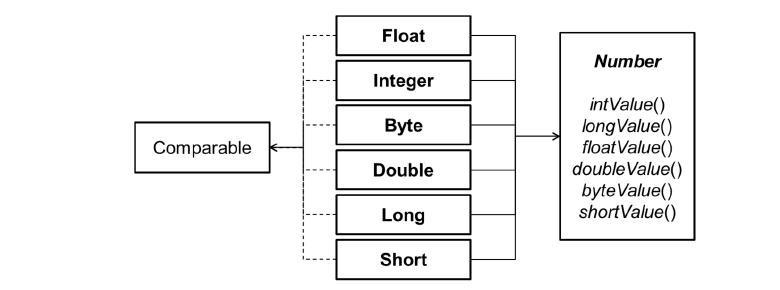
Приведение типов в выражении:

* Если одно из чисел double -> double
* Иначе, одно из чисел float -> float
* Иначе, одно из чисел long -> long
* Иначе оба в int

Операции над byte, short,char имеют тип int.Поэтому нужно явное приведение типов для обратного типа.

**Классы-оболочки**

Объекты этих классов являются константными.



Автоупаковка – извлечение из объекта оболочки значения базового типа. (Ресурсоемкий процесс)

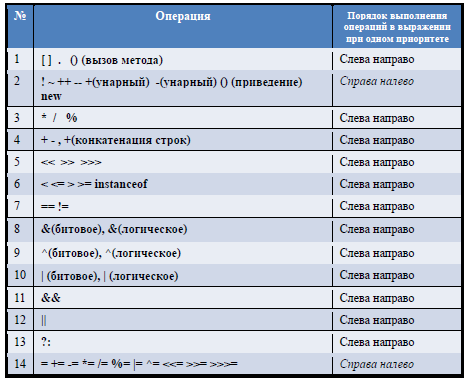
Автораспаковка – обратная операция.

*В Long,Integer, Short,Byte внутренний кеш ссылок на значения от -128 до 127.*

С помощью ссылки на класс Number возможно создание объекта и массива, сохраняющего различные типы.

**INSTANCEOF()** возвращает true , если ссылка слева приводима(расширяема) до ссылки справа.

Приоритеты операций



*Модификатор Strictfp включает режим строгой совместимости. Результаты идентичны на любом процессоре.*

*Java.lang.Math && java.lang.StrictMath*

Статический импорт : import static java.lang.Math.PI;

**Перевод чисел в строки и обратно**

String -> int метод parseInt() , класса Integer

String -> double метод parseDouble() , класса Double

Int или double в String через метод valueOf() класса String

*Кроме того, toString(), toBinaryString(), toHexString(), toOctalString();*

**Методы**

Спецификаторы доступа:

Static public friendly synchronized

Final private native

Protected abstract strictfp

**Поля**

Спецификаторы доступа:

Static public final private

Protected friendly transient volatile

**Конструкторы**

Конструктор – специальный метод класса

2 свойства: вызов только с new , не наследуется.

**Передача параметров в методы**

ТОЛЬКО ПО ЗНАЧЕНИЮ

Память выделяется под параметры метода, и те переменные, которые являются ссылочными аргументами инициализируются значением своих фактических параметров. Т.о. , минимум 2 ссылки указывают на объект.

Параметры : фактические и формальные

При передачи в метод аргумента-ссылки, его исходное значение меняется в случае возврата из метода, чтобы этого избежать, можно применить константный объект-ссылку (например объект класса String).

**JavaBeans**

- Технология разработки многократно используемых программных компонент.

Все объекты, которые хранят данные должны применять технологию JavaBeans

Особенности :

- поля делаем private,

- создание методов get(), set() на эти поля,

- переопределение методов equals(), hasCode(), toString();

**Массивы**

- ссылочный тип. Индексация с нуля.

Значение неиниц. массивов по умолчанию – 0 (или null для ссылок);

Int[] price = new int[10];

Int[] price = new int[] {1,2,3};

Многомерные массивы – массивы массивов

Int[][] price = new int[10][5];

Int[] price1 = new int[2][];

Price1[0]=new int[10];

Price1[1]=new int[2];

Length(), clone(), + все методы класса Object

**ООП**

**Инкапсуляция** – разделение внутренней реализации и интерфейса

**Наследование**(Обобщение типа) –

2 синтаксические конструкции : extends и implements

Логически позволяют обобщать типы:

Наследование – реализация отношения «есть».( Стол-мебель)

Интерфейсы - реализация заключения контракта на определенное действие. При имплиментации определяются все действия четко по контракту.

Обобщение типа позволяет выделить общие хар-ки.

**Полиморфизм** – многоформие.

Есть 1 интерфейс и много его реализаций. Пример – переопределение метода equals

Возможность использовать 1 имя для нескольких реализаций. При использовании нужно определить правила выбора этой возможности.

Перегрузка. Разрешить перегрузку возможно по типу ссылки.

**Динамическое связывание**

– замещение поведения в зависимости от типа. (переопределение метода). Пример о буфете и автомате.

**Наследование**

Все объекты всех классов создаются с помощью конструкторов. Чтобы создать объект производного класса, должна создаться его наследуемая часть, она создается с помощью конструктора. Поэтому, при выполнении конструктора производного класса, изначально создается конструктор базового.

Вызов this() и super() в одном конструкторе быть не может.   
super.method() – метод базового класса.

**Переопределение методов.**

- метод с той же сигнатурой.

Методы определяют поведение объектов.

*Например : 2 объекта : человек (методы кушать(), лечить())*

*Доктор(методы кушать(), лечить())*

**Статические методы:**

Полиморфизм неприменим. Т.к. cтатические методы не описывают поведение объекта  
Замещения нет, но в производном можно написать такой же static метод.

Наследуются статические методы? Технически да, главное , чтобы атрибут доступа позволил.

Но наследование как поведение не происходит, т.к. это не поведение.   
Поэтому, переопределяются нестатика, а перегружаются, все что угодно.

**Методы подставки:**

Возможность указывать другой тип возвращаемого значения при переопределении методов (ниже по иерархии классов).

**Приведение ссылочных типов:**

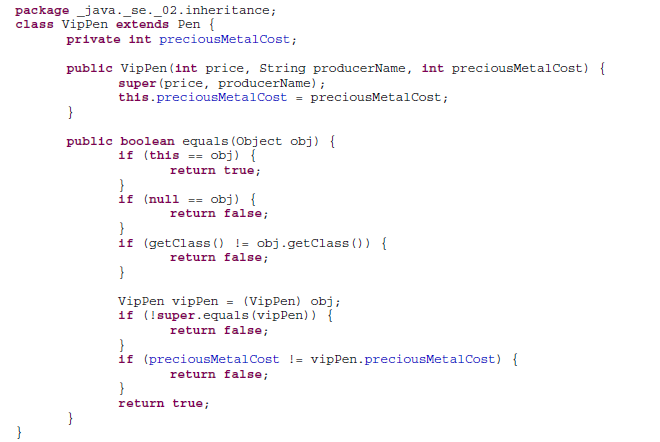
Ссылка базового типа ob1 = new ссылка производного типа();

**Перегрузка методов**

Одинаковое имя, но различный список параметров и возвр. Значения.

Для предотвращения переопределения в сигнатуре метода указать final

**Переопределение метода equals**



**Абстрактные методы и классы**

При расширении абстр. Класса все его абстр. Методы необходимо переопределить, иначе класс-наследник также объявить абстрактным.

Нельзя создавать объекты абстрактыных классов, однако можно объявлять абстрактные переменные.

Конструкторы в абстрактных классах есть, но вызывать через new нельзя.

Для запрета создания объектов любого класса можно сделать private конструктор.

**Класс Object**

clone() – битовая версия объекта

equals() – сравнение ссылок на объекты

finalize() – не используется, для рекомендации по очистке ресурсов . Обычно, объект с данным методом удаляется за 2 прихода сборщика мусора.

hashCode() – уникальный числовой код объекта.

getClass() – в пермаментной памяти создается объект типа Class с метаинформацией о нашем объекте. Этот объект содержит метаданные.

toString() – информация о пакете в котором находится класс + само имя класса + поля объекта и их значения.

**Интерфейсы**

public friendly

реализуют отношение «умеет делать»

по умолчанию методы *public abstract*

поля *final public static*

*при реализации интерфейсов, нужно явно прописать модификатор доступа public для методов*

(т.к. при переопределении расширение видимости возможно, а сужение нет)

Нельзя создавать объект интерфейса, только ссылки интерфейсного типа на объекты, которые реализуют данный интерфейс.

InstanceOf применим и для интерфейсов  
Интерфейсы можно вкладывать в другие интерфейсы, например, Map.Entry

**Интерфейс Clonable**

Для создания объекта с таким же состоянием необх. Реализовать данный интерфейс т.к. метод clone() класса Object имеет атр доступа protected.

Не содержит методов и относится к помеченным tagged.

Его реализация гарантирует , что метод clone() класса Object возвратит точную копию вызвавшего его объекта с воспроизведением значний всех его полей.

В противном случае CloneNotSupportedException

*Данный интерфейс очень затратный , лучше написать отдельный класс(FactoryClone)*

**Интерфейс Comparable**

Для объектов , которые обладают свойством естественной сортировки.

*Int compareTo(E other)* сравнивает this и переданный объект.

Возвращает отрицательное значение,если this<other, 0 – равны, положительное если,this>other

**Интерфейс Comporator**

При реализации данного интерфейса существует возможность сортировки списка объектов конкретного типа по правилам, определенным для этого типа.

Для этого необходимо реализовать метод *int compare(T ob1, T ob2)*

Java.util.Comporator содержит 2 метода:

*int compare(T ob1, T ob2)*

*Boolean equals(Object obj)* true если obj это Comporator и у него такой же принцип сравнения

**Перечисления**

* Типизированный, безопасный способ задания фиксированных наборов значений. Класс специального вида, не имеет наследников, сами наследуются от Java.lang.Enum и реализуют Comparable, Serializable.

Не могут быть абстрактными и содержать абстрактные методы, но могут реализовывать интерфейсы.

Нельзя создать с помощью new. Каждый объект уникален и создается при загрузке перечисления в виртуальную машину.

Как и обычные классы могут реализовывать поведение, содержать вложенные классы.   
По умолчанию, элементы перечисления public static final

Методы( values(), valueOf(arg), toString(), equals(obj), ordinal())