Контрольная работа № 1.

1. Перечислите атрибуты доступа, объясните их действие.

Атрибуты доступа дают возможность управлять инкапсуляцией на уровне классов. Атрибутов доступа 4:

- *private* – поля и методы доступны только в классе, где они описаны (использование метода/поля в точке его написания). Для организации доступа к private полям вне класса используются специальные методы (геттеры и сеттеры). С атрибутом *Private* могут быть объявлены методы, поля. Классы и интерфейсы не могут быть *private*.

- *friendly* (по умолчанию, *private-package*) – инкапсуляция на уровне пакета. Поля и методы доступны в классе, где они объявлены и в других классах в том же пакете. С атрибутом *Friendly* могут быть объявлены классы, поля и методы.

- *protected* – поля и методы доступны в классе, где они объявлены, в других классах в том же пакете, в классе-наследнике (производном классе), в том числе и в другом пакете. С атрибутом *protected* могут быть объявлены поля и методы, классы (внутренние).

- *public* – поля и методы доступны в классе, где они объявлены, в других классах в том же пакете и вообще везде. С атрибутом доступа *public* могут быть объявлены поля, методы, классы.

2. Что такое пакеты в java-программе, что представляют собой пакеты на диске? Каково соглашение по именованию пакетов? Как создать пакет?

Пакеты в java-программе = каталоги в файловой системе. Они нужны для структурирования (иерархии), а также создания уникальности имен классов (исключение конфликтов имен). Пакет прописывается при помощи ключевого слова *package*. Имена пакетов отделяются точкой (java.util.Date)

Правила именования пакетов:

-имена пакетов пишутся с маленькой буквы;

- если необходимо использовать 2 слова, то они записываются с использованием snake case (два_слова);

- традиционно используют 4-8 пакетов, начиная с домена компании, записанного наоборот (by.epam.);

- все пакеты именуют в единственном числе.

Пакеты позволяют сделать другой вид инкапсуляции (архитектурная), где можно нарушать область видимости.

3. Объясните, какие классы, интерфейсы, перечисления необходимо импортировать в вашу программу, как это сделать. Влияет ли импорт пакета на импорт классов и др., лежащего в подпакетах? Какой пакет в Java импортируется по умолчанию?

Импорт классов, интерфейсов, перечислений зависит непосредственно от поставленной перед разработчиком задачи.

Импорт осуществляется при помощи ключевого слова *import* перед объявлением класса.

При импорте пакета нужно отдельно подключать его подпакеты. Несмотря на то, что есть возможность импортировать сразу все подпакеты (java.util.*), необходимо осознавать, что такой подход затрудняет чтение кода другими разработчиками.

Пакет java.lang импортируется по умолчанию, содержит класс System.

В случае импорта двух классов с одинаковыми именами, лежащих в разных пакетах (Java.util.Date и java.sql.Date) импортировать можно только 1 класс, к другому придется обращаться по полному имени, но рекомендуется обращаться к каждому по полному имени, чтобы не возникало путаницы.

Импорт упрощает способ обращения к методам и полям класса.

4. Объясните различия между терминами “объект” и “ссылка на объект”.

Объект – экземпляр класса, которые содержит описание свойств и поведения. При вызове поведения объекта могут меняться значения его свойств, а при изменении свойств – поведение.

Ссылка на объект – ячейка, которая хранит адрес объекта, но не сам объект. Она не изменяет объект и никак не влияет на его свойства и поедение.

Scanner scanner (*ссылка на объек*т) = new scanner() (*создание объекта*);

Объекты создаются в heap (ключевое слово *new* выделяет память под объект), а локальные переменные в stack (хотя могут и в heap).

Ссылка имеет тип и объект имеет тип, в большинстве случаев они будут различные.

5. Какие примитивные типы Java вы знаете, как создать переменные примитивных типов? Объясните процедуру, по которой переменные примитивных типов передаются в методы как параметры.

Примитивные типы – типы данных, которые описываются одним значением*!* Ссылочные типы так описать невозможно (это комплексное значение).

Примитивные типы нельзя использовать всегда, например, в Коллекциях, их нужно обернуть в класс-оболочку.

Вычисления с примитивными типами выполняются быстрее в несколько десятков раз, т.к. выполняются процессором!

Примитивных типов в Java 8 шутк:

Byte (8 бит), short (16 бит), int (32 бита), long (64 бита),

float (32 бита), double (64 бита),

char (16 бит),

boolean (1-2 бита).

При создании переменной нужно определиться с ее типом (тип определяет, какие значения может хранить переменная) и именем переменной (в соответствии с Code convention).

При объявлении переменной ей присваивается значение по умолчанию (0, 0.0, ‘’, false).

int i; double z; char = ‘a’;

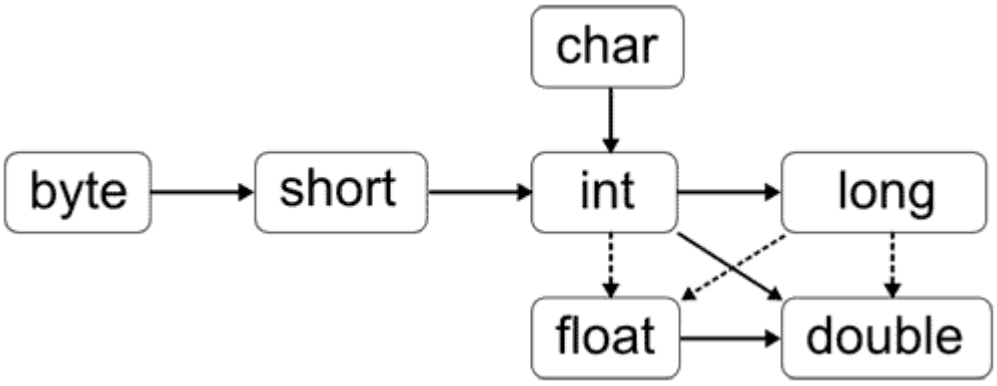
Переменные типа int, char хранятся в одном месте, символы закодированы в таблице Юникод в 16-ричной системе счисления.

В Java все типы данных знаковые, т.о. 1-ый бит хранит в себе информацию о знаке (+/-).

Примитивные и ссылочные типы передаются в методы **по значению!**

6. Объясните, что такое явное и неявное приведение типов, приведите примеры, когда такое преобразование имеет место.

Неявное приведение типов – преобразование одного типа к другому, которое происходит при участии компилятора в соответствии со спецификациями языка java. Такое преобразование компилятор выполнит самостоятельно без участия разработчика (например, в переменную с большим диапазоном значений записать переменную с меньшим диапазоном значений).



Штриховкой обозначены переходы при которых возможна потеря точности вычислений.

При переходе с int, long во float, double теряется точность вычислений!!!

При неявном приведении типов java все сделает автоматически.

Явное приведение типов – преобразование одного типа к другому, при котором возможна потеря данных и нарушение точности вычислений (например, в переменную с меньшим диапазоном значений записывается переменная с большим диапазоном). Компилятор отслеживает такие вычисления и требует участия разработчика. Т.о. при явном приведении типов вся ответственность ложится на разработчика.

Для контроля точности вычислений пишутся специальные численные методы или используются классы-оболочки (как в java).