1. **Объясните что имеется ввиду когда говориться Java язык программирования или Java платформа**

Java язык – ООП язык программирования, статически типизируемый.

Java как платформа – состоит из Java API (библиотеки Java), Java Virtual Machine (JVM). Jvm компилирует код программиста в .class файлы, которые являются “Intermediate language” между кодом юзера и машинным (00110101) кодом.

1. **Поясните как связаны имя java-файла и классы, которые в этом файле объявляются.**

Имя файла должно совпадать с public class в этом файле. Каждый класс хранится в отдельном файле. Регистр также важен.

1. **Расшифруйте аббревиатуры JVM JDK JRE, покажите где они находятся и что собой представляют**

*The Java Development Kit (JDK) is a software development environment used for developing Java applications and applets. It includes the Java Runtime Environment (JRE), an interpreter/loader (Java), a compiler (javac), an archiver (jar), a documentation generator (Javadoc) and other tools needed in Java development.*



JVM – виртуальная машина Java. Используется для преобразования кода. JRE – среда разработки, включает в себя JVM и минимальный пакет для разработки. JDK – включает в себя JRE и добавляет инструменты для разработчика (интерпретатор, загрузчик, компилятор, архиватор (jar) и тд.

1. **Объясните как скомпилировать и запустить приложение из командной строки, а также зачем в переменных среды окружения прописывать пути к jdk.**

Написать код в файле с расширением Java. Наименование класса должно совпадать с именем. Например файл Hello.java с классом Hello и методом (public static void main). Можно запустить при помощи:

Javac Hello.java – компиляция программы, создание Hello.class

Java Hello – запуск программы Hello.class. (Запустится метод main).

Путь к JDK прописывается в переменные среды, чтобы операционная система получила настройки среды/окружения. Переменная Path содержит список директорий, который ОС автоматически проверяет, если юзер не указал явно путь к нужному исполняемому файлу.

Можно прописать в windows или устанавливать при каждом запуске командной строки (если устанавливать в cmd через set то установится только для этой сессии)

В переменную Path для java прописывается путь к папке jdk\bin

1. **Перечислите атрибуты доступа, объясните их действие.**

Модификаторы доступа Java:

Public – доступ из любой части программы

Protected – доступ из всего пакета, плюс возможен доступ из другого пакета при наследовании класса

Default – доступ из всего пакета, доступ извне при помощи наследования невозможен

Private – доступ только внутри класса

1. **Что такое пакеты в Java программе, что представляют собой пакеты на диске? Каково соглашение по именованию пакетов. Как создать пакет?**

Package используется для каталогизации классов в программе Java. Также помогает избежать дублей наименований классов. На диске пакеты = директории, то есть если класс содержит package by.epam.homework то в корне директории должны быть созданы папки by > epam > homework.

Полное имя класса Main будет by.epam.homework.Main, можно не указывать полностью если написать import by.epam.homework.Main.

Наименования пакетов пишутся нижним регистров, начинаются обычно с домена, далее наименование компании, далее тема (подраздел).

1. **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 Java импортируется по умолчанию.**

По умолчание стоит импорт Java.lang

Для множественного импорта можно использовать символ «\*». Например import java.util.\* импортирует весь пакет util. Также если нужно можно импортировать статически, например import static java.lang.Math.\* чтобы использовать вместе Math.sqrt() просто sqrt().

При одинаковых именах и импортах обоих пакетов придется указывать полное имя например new Java.util.Date();

1. **Объясните различия между терминами «объект» и «ссылка на объект».**

Cat cat = **new** Cat(); // создается объект Кот и ссылка на него  
Cat cat1 = **new** Cat(); // создается еще один объект Кот и ссылка на него  
cat = cat1; // первой ссылке присваивается значение второй, то есть теперь первая ссылка ссылается на второй объект кот. Фактически к второму объекту мы можем получить доступ из двух ссылок.

cat1.**name** = **"vaska"**; // присваивание объекту значение name;  
System.***out***.println(cat1.**name**); // выводим значение name объекта, на которого ссылается ссылка cat1  
System.***out***.println(cat.**name**); // выводим значение name объекта, на которого ссылается ссылка cat

Объект, который мы создали в первой строчке больше не содержит ссылок на себя, а значит будет собран garbage коллектором (подсвечен IDE как неиспользуемый).  
  
**int** x;  
**int** y = 5;  
x = y;  
y = 6;  
System.***out***.println(x);  
System.***out***.println(y);

С примитивами иначе. Здесь x хранит значение, присвоеное ему переменной y. Далее мы изменяем значение y, однако значение x остается не изменным, т.к. оно хранит не ссылку на y, а именно значение (5).

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

Примитивы Java: byte, short, int, long, float, double, Boolean, char.

***Java передает параметры по значению. Всегда.*** *Это означает — "скопировать значение и передать копию."*

При передаче примитива в метод все понятно. Передается копия например числа, теперь то что происходит в методе с числом никак не влияет на «внешнее» число.

При передаче ссылки например void feedCat(vaska) передается копия содержимого, т.е. копия ссылка на объект кота. То есть все, что будет «сказано» сделать ссылке внутри метода принесет изменения в тот же внешний объект.

1. **Каков размеры примитивных типов, как размер примитивных типов зависит от разрядности платформы, что такое преобразование (приведение) типов и зачем оно необходимо? Какие примитивные типы не приводятся ни к какому другому типу.**

Byte от -128 до +127

Short от -32768 до + 32767

Int – 32 бита, от -2^31 до +2^31-1

Long – 64 бита, от -2^63 до +2^63-1

Float / double – 32 / 64 бита, числа с плавающей точкой.

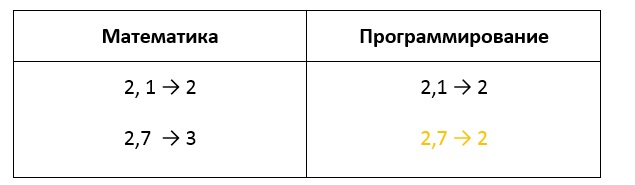
Boolean / char.

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

Приведение типов происходит неявно от меньшего к большему и явно наоборот. Например:

**int** x = 10;  
**long** y = x;  
  
**long** t = 10;  
**int** d = (**int**)t;

Здесь все логично, т.к. если значение x поместилось в ИНТ, то логично что оно поместится и в более «широкий» тип, поэтому явное преобразование не требуется. Если просиходит преобразование наоборот, из большего к меньшему, требуется явное указание.



1. Что такое литералы в Java программе, какую классификацию литералов вы знаете, как записываются литералы различных видов и типов в Java программе?

Литерал – явно заданное значение переменной.

*Все литералы — это примитивные значения (строки, числа, символы, булевы значения). Нельзя создать литерал-объект. Единственный литерал, связанный с объектом — это null. В соответствии с примитивами, литералы также делятся на разделы:*

1. *Числовые:*
   1. *Целочисленные;*
   2. *С плавающей точкой;*
2. *Строковые;*
3. *Символьные;*
4. *Логические.*
5. **Как осуществляется работа с типами при вычислении арифметических выражений в Java?**

*Знаки операции инкремента и декремента могут размещаться как до, так и после переменной. Эти варианты называются соответственно* ***префиксной*** *и* ***постфиксной*** *записью этих операции.*

*Знак операции в префиксной записи возвращает значение своего операнда* ***после*** *вычисления выражения.*

*При постфиксной записи знак операции сначала воз­вращает значение своего операнда и только* ***после*** *этого вычисляет инкремент или декремент, например:*

***int*** *x = 1, y, z;*

*y = ++x;*

*z=x++;*

*Переменная y будет присвоено значение 2, поскольку сначала значение x будет увеличено на 1, а затем результат будет присвоен переменной y. Переменной z будет присвоено значение 1, поскольку сначала переменной z будет присвоено значение, а затем значение x будет увеличено на 1. В обоих случаях новое значение переменной x будет равно 2.*

<https://javarush.ru/groups/posts/1388-operacii-nad-primitivnihmi-tipami-v-java>

При сложении операции расширения происходят автоматически. При сужении обязаны указать явно.

Арифметические операции над мелкими типами автоматически расширятся до int, далее расширяется только по мере надобности

Byte < short < int < long < float < double

В операции 5 + 15 + 5L + 15F + 1.08

Первое сложение будет в типе инт, вторая операция расширится до лонг

При сужении необходимо указать явно например: int x = (43.15);

При этом дробная часть будет не округлена, а отброшена.

1. **Что такое классы оболочки, для чего они предназначены? Объясните что значит объект класса оболочки – константный объект?**

Классы оболочки представляют собой объекты и предназначены для расширения возможностей работы с примитивами (теоретически при надобности можно написать класс-оболочку на любой объект, если понадобиться поменять связь команда->класс, на команда->класс-оболочка-> класс)

Например int x = 10; можно заменить на Interger x = 10; тогда у нас появятся доступны метода класса Integer. Класс оболочку имеет каждый примитив.

Объект класса оболочки является константный объектом означает, что Integer x = 10; имеет private final int поле и если мы присвоим x другое значение, то под капотом будет создан другой объект x = new Integer (20), то есть поле x immutable;

Также immutable является любое поле String;

1. **Объясните разницу между примитивными и ссылочными типами данных. Поясните существующие различия при передаче параметров примитивных и ссылочных типов в методы. Объясните как константный объекты ведут себя при передаче в метод**

Существует два типа данных в Java:

* простые или примитивные типы данных;
* ссылочные типы данных (ссылка/объект)

Примитивы создаются при помощи ключевых слова int x; и могут инициализироваться литералами например int x = 15;

Ссылочные создаются консутрукторами например User user = new User();

Параметры передаются по значению. Всегда. При передаче примитива это сразу понятно, при передаче ссылки например

Cat A = **new** Cat ();

doStuff (А);

**void** doStuff (Cat B) {

B = **new** Cat (); / / Не повлияет на ссылку A

}

Передается копия ссылки. То есть объект остается один и он же меняется, но ссылка становится другой. Аналогия – создания пульта управления к ТВ. Два пульта управляют одним и тем же тв.

1. **Поясните что такое автоупаковка и автораспаковка.**

**Автоупаковка**

Integer integer = 9;

**Распаковка**

**int** in = 0;

in = **new** Integer(9);

Автоупаковка происходит автоматически при передаче примитива в метод, если этот метод ожидает класс оболочку. Или когда значение присваивается переменной, ожидающей значений класса оболочки. В обратных случаях, когда передается оболочка, а ожидается примитив – происходит распаковка.

Автоупаковка требует явного приведения, если оно (приведение) необходимо.

**17.Перечислите известные вам арифметические, логические и битовые операторы, определите случаи их употребления. Что такое приоритет оператора, как определить в какой последовательности будут выполнять операции и выражения, если несколько из них имеют одинаковый приоритет?**

Арифметические**: + - \* / %**

Сравнение**: > < >= <= == !=**

Присваивание: **=** Работает справа налево

Унарные операции типо **x++** и **++x. ++x** - ПРЕфиксная операция, то есть сначала происходит сложение, после – присваивание.

**X++** - ПОСТ-фиксная операция, то есть сначала будет присваивание, после – сложение.

Операторы **|| &&** ! для булевых сравнений.

Побитовые операции:

* *Побитовые операции рассматривают исходные числовые значения как поля битов и выполняют над ними следующие действия:*
* *установка бита в i-ой позиции поля результата в 1, если оба бита в i-ых позициях операндов равны 1, или в 0 в противном случае – побитовое И ("&");*
* *установка бита в i-ой позиции поля результата в 1, если хотя бы один бит в i-ых позициях операндов равен 1, или в 0 в противном случае – побитовое ИЛИ ("|");*
* *установка бита в i-ой позиции поля результата в 1, если биты в i-ых позициях операндов не равны друг другу, или в 0 в противном случае – побитовое исключающее ИЛИ ("^");*
* *сдвиг влево битов поля первого операнда на количество битов, определяемое вторым операндом (бит знака числа при этом не меняется) – побитовый сдвиг влево с учетом знака "<<";*
* *сдвиг вправо битов поля первого операнда на количество битов, определяемое вторым операндом (бит знака числа при этом не меняется) – побитовый сдвиг вправо с учетом знака ">>";*
* *сдвиг вправо битов поля первого операнда на количество битов, определяемое вторым операндом (бит знака числа при этом также сдвигается) – побитовый сдвиг вправо без учета знака ">>>".*
  1. **Укажите правила выполнения операций с плавающей точкой в Java согласно стандарту IEEE754. Как определить что результатом вычисления стала бесконечность или не число?**

Подобно тому как в десятичной системе нельзя точно записать 1/3, в десятичной нельзя точно записать 1/10 (поэтому для точных вычислений лучше использовать класс BigDecimal).

У float и double выделено по 32 и 64 бита соответственно, значит double имеет большую точность вычислений.

Можно использовать Math.abs – param – где param – заданное нами значение, необходимое для точности (например 0,0001)

Вещественное число представляется 3 числами: знак (+ или -), экспонента и мантисса.

*Для заданных целых чисел S, C, Q значением соответствующего вещественного числа является: (−1)s × c × bq , где b является основанием (2 или 10).* *Например, число с основанием 10, битом знака 1 (число отрицательное), мантиссой 12345 и экспонентой −3 определяют число (−1)1 × 12345 × 10−3 = −12.345.*

* 1. **Что такое статический импорт, какие элементы можно использовать при статическом импорте?**

Первое – это возможность статического импорта классов. Import static Math. Например позволяет вызывать методы напрямую без указания класса, что упрощает запись и чтение кода.

Например static import Java.lang.System.out.

Позволяет обращаться напрямую и вызывать: out.println(); вместое System.out.println;

Не следует злоупотреблять статическим импортом дабы избежать конфликта имен и сохранить понимаемость кода.

Можно применять для статических членов класса или интерфейса.

* 1. **Объясните работу операторов if switch while do-while, for, for-each**

Оператор IF самый простой и выполняет команды если условие в скобках возвращает булевое тру.

Например:

**boolean** condition = **true**;  
  
**if**(condition){  
 *doSmth*();  
}

Может испольоваться с связкой else (ЕСЛИ), если нужно указать что делать если условие не соблюдается.

Switch является аналогом if-else конструкций и преднозначен для применения в случаях громоздких условий для упрощения кода:

**switch** (condition){  
 **case (**1): *doSmth*(); **break**;  
 **case (**20): *doSmth*();   
 **case (37**): *doSmth*(); **break**;  
}

Здесь в condition мы ожидаем значение типа инт и в зависимости от его делаем определенные действия. Break заканчивает цикл, если он не указан (как в случае 20) , то будут выполнены все действия пока не встретится оператор break. То есть в примере при cond = 20; метод будет вызван два раза.

Операторы while и do/while выполняют цикл пока верны условия и отличаются последовательностью выполнения:

**boolean** condition = **false**;  
  
**do** {  
 *doSmth*();  
}  
**while** (condition);

**////////////**

**while** (condition){  
 *doSmth*();  
}

В первом случае метод будет выполнен один раз перед сравнением, после condition вернет false и цикл завершится, то есть метод вызовут один раз. Во втором случае метод не будет вызван ни разу.

Цикл for:

**for**(**int** i = 0; i < 10; i++){  
 *doSmth*();  
}

Сначала проверяет условие, после выполняет тело метода, в конце производит операцию после второй запятой, в примере это I = I + 1;

For-each разновидность цикла for для упрощенного прохождения по всем (по каждому дословно) элемента массивов.

Пример:

**for**(String element: array){  
 *doSmth*();  
}

Создаем переменную element типа String, которой будет по очередно присваиваться значение каждого элемента в массиве array. Далее можно делать нужные методы применяя каждый элемент массива.

Для изменения элементов массивах лучше использовать классические конструкции типа:

**for**(**int** i =0; i < array.length; i++){  
 array[i] = 55;  
}

* 1. **Объясните работу операторов Instanceof. Что будет результатом работы если слева от него будет стоять ссылка null**

System.***out***.println(**null instanceof** Object);

Всегда вернет false;

x **instanceof** y - определяет является ли объект, на который ссылается x потомком класса y. Возвращает булевое значение.

Как определяется? При вызове любого класса-потомка автоматически сначала вызываются все конструкторы базовых классов, потому если x потомок от y, то при создании x были использованы конструкторы класса y.