Basics of software code development

1.Объясните, что имеется в виду, когда говорится: Java-язык программирования и Java-платформа.

Java как язык программирования является высокоуровневым, статически-типизированным и объектно-ориентированным.

Java как платформа — это программное обеспечение, представляющее собой рабочую среду для работы программ, написанных на Java (и не только). Она состоит из Java API и Java виртуальной машины (JVM).

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

JAVA-файл должен именоваться так же как и public class внутри него. Несоблюдение этого соглашения может стать причиной многих проблем, которые выявятся во время компиляции.

3.Расшифруйте аббревиатуры JVM, JDK и JRE; покажите, где "они находятся" и что собой представляют.

- JDK (Java Development Kit) - предоставляет среду для разработки и выполнения (запуска) Java-программы. Включает в себя исполнительную среду JRE, компилятор java-кода в байт-код, debugger-отладчик ошибок, disassembler.
- JRE (Java Runtime Environment) - это исполнительная среда Java в которой выполняются программы, написанные на этом языке. Среда состоит из виртуальной машины – Java Virtual Machine(JVM) и библиотеки Java классов.
- JVM (Java Virtual Machine) -основная часть исполняющей среды JRE. Виртуальная машина Java интерпретирует и исполняет байт-код Java.

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

В командной строке выбрать директорию расположения файла программы. Компиляция программы из командной строки: javac filename.java. На выходе получаем filename.class содержащий байт-код. Запуск программы из командной строки: java filename.

Переменные среды нужны для запуска программ через командную строку, в переменных среды прописывается путь к JDK.

5. Перечислите атрибуты доступа, объясните их действие.

private - наиболее строгий модификатор доступа. Члены класса доступны только членам данного класса;

default - модификатор доступа по умолчанию. Члены класса доступны классам, находящимся в том же пакете;

protected – модификатор доступа. Члены класса доступны классам, находятся в том же пакете и подклассам в других пакетах;

public – модификатор доступа. Члены класса доступны для всех классов в этом и других пакетах, т.е. доступны везде.

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

Пакет – это набор классов java, объединенных по каким-либо логическим признакам, темой, свойством. Пакеты на диске представлены в виде папок. Чтобы указать, что класс принадлежит определенному пакету, надо использовать директиву package, после которой указывается имя пакета (package name_ package). Имена пакетов должны быть в нижнем регистре.

7. 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?

Если надо использовать классы из других пакетов, то необходимо подключить эти пакеты и классы. Исключение составляют классы из пакета java.lang, которые подключаются в программу автоматически.

Чтобы импортировать пакет необходимо перед объявлением класса произвести импорт:

import PackageName.ClassFileName

PackageName – имя пакета

ClassFileName – имя файла с расширением .java, через точку могут перечисляться по иерархии подпакеты.

import PackageName.* - импорт всего пакета PackageName.

После импорта пакета или класса нет необходимости в программе указывать полный путь к классу или объекту класса.

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

Объект — это экземпляр класса, создается при помощи кодового слова new. Ссылка хра-нит адрес ячейки памяти, в которой хранится объект.

Создание объекта: Class variable = new Class (attribute)

variable – это ссылка на объект.

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

1. Целочисленные типы:
   - byte: byte a = 6;
   - short : short b = 50;
   - int : int x = 128;
   - long : long y = 300;

2.Числа с плавающей точкой:
   - float : float f = 5.9f;
   - double : double d = 53.86;

3. Символьные типы:
   - char : char c = ‘s’;

4. Логические типы:
   - boolean : boolean b = true; boolean b1 = false.

Если методу передается аргумент примитивного типа, то происходит передача по значению. То есть, делается копия аргумента.

10. Каков размер 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.

1. Целочисленные типы:
   - byte (1 байт, [-128, 127]);
   - short (2 байта, [-32768, 32767]);
   - int (4 байта, [-2147483648, 2147483647]);
   - long (8 байт, [-9223372036854775808,9223372036854775807]);

2.Числа с плавающей точкой:
- float (4 байта, [-3.4E+38, 3.4E+38]);
- double (8 байт, [-1.7E+308, 1.7E+308]);

3. Символьные типы:
- char (символ Unicode, 2 байта, [0, 65536]);

4. Логические типы:
- boolean (true, false).

Размер типов не зависит от разрядности системы.

В Java возможны преобразования между целыми значениями и значениями с плавающей точкой. Кроме того, можно преобразовывать значения целых типов и типов с плавающей точкой в значения типа char и наоборот, поскольку каждый символ соответствует цифре в кодировке Unicode. Тип boolean является единственным примитивным типом в Java, который нельзя преобразовать в другой примитивный тип. Кроме того, любой другой примитивный тип нельзя преобразовать в boolean.

Т.к. разные типы данных занимают разный объем памяти, то иногда необходимо преобразование одного типа данных в другой (явное приведение).

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

Преобразование типов в Java бывает двух видов: неявное и явное.

Неявное преобразование типов выполняется в случае если выполняются условия:
- Оба типа совместимы
- Длина целевого типа больше или равна длине исходного типа

Во всех остальных случаях должно использоваться явное преобразование типов.

Пример явного преобразования:

```
int a;
byte b = (byte) a; // Принудительное преобразование int в byte
```

Пример неявного преобразования:

```
int a;
long x = a;
```

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

Любая константа в Java является литералом. Все литералы — это примитивные значения (строки, числа, символы, булевы значения). Нельзя создать литерал-объект. Единственный литерал, связанный с объектом — это null.

Классификация литералов:

- Целочисленные литералы:

Данный вид литералов-самый простой. Числа записываются в их стандартном виде без указывающих символов и прочего. Любое целое число-по умолчанию целочисленный литерал.

```
int a = 12;
```

- Литералы с плавающей точкой:

Числа с плавающей точкой, или дробные числа, могут записываться двумя способами. Первый-в качестве классической десятичной дроби: 3.14. Второй- обычная десятичная дробь плюс суффикс в виде символа e или E и степени 10, на которую нужно умножить указанную дробь

По умолчанию тип литерала определяется как double. Для записи типа float в конце записи необходимо добавить f или F.

```
double f=3.14;          // вывод 3.14
double y=2.3E-2;        // вывод 0.023
```

- Символьные литералы:

Символьные литералы в Java представлены кодовой таблицей Unicode, то есть каждый символ-16-битовое значение. Для обозначения символа в коде его обособляют одинарными кавычками.

Символы, которые нельзя просто так ввести с консоли, можно задать в их 16-битовом виде. Для этого необходимо указать код символа с префиксом '\u'. Также символы можно указывать в восьмеричном стиле (трехзначный номер), добавив в начале просто обратный слеш.

```
char x='%';          // вывод %
char y='\u00f8';     // вывод Ø
char z='\101';       // вывод A
```

- Строковые литералы:

Строковые литералы — это набор символов, заключенных в двойные кавычки.

```
String str = “Hello, World”;          // вывод Hello, World
```

- Логические литералы:

Имеется всего 2 значения: false и true.

```
boolean a = true;
boolean b = false;
```

13. Как осуществляется работа с типами при вычислении арифметически выражений в Java?

В Java определены следующие арифметические бинарные операции:

- сложение "+";
- вычитание "-";
- умножение "*";
- деление "/";
- вычисление остатка от деления целых чисел "%".

Унарные операции:

- инкремент "++" (только для целых) – увеличивает значение переменной на 1;
- декремент "--" (только для целых) – уменьшает значение переменной на 1.

При вычислении арифметических выражений тип конечного результата продвигается до int для меньших типов либо до double для Long. При этом при вычислении выражения с разными типами, конечный результат продвигается до большего типа среди всех переменных.

14. Что такое классы-оболочки, для чего они предназначены? Объясните, что значит: объект класса оболочки – константный объект.

Классы-оболочки Java являются Объектным представлением восьми примитивных типов в Java.

Обычные примитивы занимают меньше места в памяти и поэтому над ними можно проводить операции быстрее. Но над объектами аналогичных классов-оболочек можно производить большее количество операций и специальных действий, например, перевести текст в число (.parseInt). Эти операции, методы уже прописаны в классе-оболочке.

Константный объект - это значение примитива, для которого объект является оболочкой. К примеру, для int - оболочка java.lang.Integer для long - оболочка - java.lang.Long. В исходном коде класса java.lang.Integer имеется приватная финальная переменная типа int - private final int value, которую нельзя изменить. То есть объявляя Integer x = 1; на самом деле, выполняется нечто вроде:

Integer x = new Integer(1);

Далее, если мы захотим поменять значение x = 2, то переменная x будет указывать уже на другой объект, а не перезатрет значение предыдущего. Предыдущий объект, будет собран сборщиком мусора, если на него более не имеется ссылок.

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

Каждая переменная в Java имеет тип данных: примитивный и ссылочный.

Примитив содержит значение переменной непосредственно в памяти, выделенной для нее. Ссылка не содержит объект, но содержит ссылку на него, которая указывает на другое место в памяти, где он хранится. Через такую ссылку можно получить доступ к полям и методам ссылочного объекта.

Если методу передается аргумент примитивного типа, то происходит передача по значению. То есть, делается копия аргумента.

В отличие от переменных примитивных типов, объекты класса передаются по ссылке. Это значит, что изменения, сделанные в теле функции (методе) будут изменять также значения объекта, который задавался в качестве аргумента.

Кроме переменных, в Java для хранения данных можно использовать константы. В отличие от переменных константам можно присвоить значение только один раз. Константа объявляется так же, как и переменная, только вначале идет ключевое слово final. Константы позволяют задать такие переменные, которые не должны больше изменяться.

16. Поясните, что такое автоупаковка и автораспаковка.

Преобразования примитивных типов в эквивалентные объекты в Java реализуется автоматически. Это свойство известно как **Автоупаковка** (Autoboxing). Обратный процесс соответственно – **Распаковка** (Unboxing) т.е. процесс преобразования объектов в соответствующие им примитивные типы.

Автоупаковка: Integer integer = 9;

Автораспоковка: int in = 0;

in = new Integer(9);

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

- Арифметические. Используются для вычислений так же как в алгебре. Допустимые операнды должны иметь числовые типы. Например, использовать эти операторы для работы с логическими типами нельзя, а для работы с типом char можно, поскольку в Java тип char — это подмножество типа int. : +,-,*,/,%,++,--

- Логические. Оперируют только с операндами типа boolean. Все бинарные логические операторы воспринимают в качестве операндов два значения типа boolean и возвращают результат того же типа.: &&,||,!

- Битовые. Для целых числовых типов данных — long, int, short, char и byte, определен дополнительный набор операторов, с помощью которых можно проверять и модифицировать состояние отдельных битов соответствующих значений. Операторы битовой арифметики работают с каждым битом как с самостоятельной величиной. : & побитовое И (AND), | побитовое ИЛИ (OR), ^ побитовое исключающее ИЛИ (XOR) , >> сдвиг вправо, >>> сдвиг вправо с заполнением нулями, << сдвиг влево, ~ побитовое унарное отрицание (NOT).

Если выражение включает несколько операторов с одинаковым приоритетом, то порядком выполнения операций управляет ассоциативность операторов. Большинство операторов ассоциативны слева направо, то есть операции выполняются слева направо. Однако операторы присваивания и унарные операторы обратно ассоциативны (справа налево).

18. Укажите правила выполнения операций с плавающий точкой в Java (согласно стандарту IEEE754). Как определить, что результатом вычисления стала бесконечность или нечисло?

Язык Java поддерживает два простых типа с плавающей точкой: float и double, и их прототип - класс-оболочка Float и Double. Они основаны на стандарте IEEE 754, определяющем двоичный стандарт для двоично-десятичных чисел с 32-битовой плавающей точкой и 64-битовой плавающей точкой удвоенной точности.

Формула вычисления десятичных чисел с плавающей точкой из чисел, представленных в стандарте IEEE754:

$$F = (-1)^S \ 2^{(E-2^{(b-1)}+1)}(1+M/2^n)$$

где:

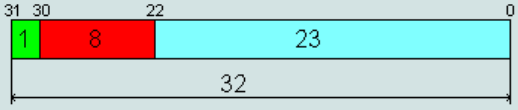
S - бит знака, если S=0 - положительное число; S=1 - отрицательное число

E - смещенная экспонента двоичного числа; exp2 = E - (2(b-1) - 1) - экспонента двоичного нормализованного числа с плавающей точкой (2(b-1) -1) - заданное смещение экспоненты (в 32-битном IEEE754 оно равно +127)

M - остаток мантиссы двоичного нормализованного числа с плавающей точкой.

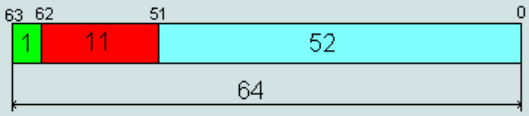
В IEEE числе с плавающей точкой выделяется 1 бит на знак, 8 бит на порядок и 23 бита на мантиссу, или дробную часть числа. Для плавающей точки с удвоенной точностью на порядок выделяется 11 бит, а на мантиссу - 52 бит.

Формат числа одинарной точности (single-precision) 32 бита:



$$F = (-1)^S 2^{(E-127)}(1+M/2^{23})$$

Формат числа двойной точности (double-precision) 64 бита:



$$F = (-1)^S \ 2^{(E-1023)}(1+M/2^{52})$$

В Java тип double имеет специальные значения для понятий «плюс бесконечность» и «минус бесконечность». Положительное число, разделенное на 0.0, дает «плюс бесконечность», а отрицательное – «минус бесконечность».

Нечисло NaN ("not a number") является результатом арифметических операций, если во время их выполнения произошла ошибка. Данное состояние может возникнуть в различных случаях, например, когда предыдущая математическая операция завершилась с неопределённым результатом, или если в ячейку памяти попало не удовлетворяющее условиям число. К операциям, приводящим к появлению NaN в качестве ответа, относятся:

- все математические операции, содержащие NaN в качестве одного из операндов;
- деление нуля на нуль;
- деление бесконечности на бесконечность;
- умножение нуля на бесконечность;
- сложение бесконечности с бесконечностью противоположного знака;
- вычисление квадратного корня отрицательного числа;
- логарифмирование отрицательного числа.

19. Что такое статический импорт, какие элементы можно импортировать при статическом импорте.

Для статического импорта используется оператор Java import static, после которого указывается полное имя класса и метод или переменная. Оператор static import можно применять для импорта статических членов класса или интерфейса. Благодаря статическому импорту появляется возможность ссылаться на статические члены непосредственно по именам, не уточняя их именем класса.

20. Объясните работу операторов if, switch, while, do-while, for, for-each. Напишите корректные примеры работы этих операторов.

- **if** - оператор условия.

Полная структура:

```
if (условие){
(выполняются действия);
} else {
(выполняются действия);
}
```

Сокращенная структура:

```
if (условие) (выполняются действия);
```

Многозвенная структура:

```
if (условие){
(выполняются действия);
} else if (условие){
(выполняются действия);
}
```

- **switch** - оператор выбора.

Без вложенности:

```
switch (выражение, переменная для сравнения с case (как правило целое число)) {
case 1: выполняются действия;      //проверяется совпадение с 1, если совпадает, то выполняется
                                   //команда, если нет, то сравнение продолжается
break; //при условии совпадения происходит выход из оператора switch
case 2: выполняются действия; //по аналогии с case1
break;
default: выполняются действия; //если ни один вариант не подошел, то срабатывает default, если его не
                               //написать произойдет ошибка
}
```

С вложенностью:

```
switch(выражение 1){
case 1: switch(выражение 2){
case 20: выполняются действия;
break;
…}
break;
}
```

- **while** - оператор цикла. Пока условие истинно выполняем

```
while (условие){выполняются действия;} //может не выполниться ни разу!
```

- **do while** - оператор цикла. do-while отличается от while только тем, что тело цикла выполнится хотя бы один раз, т.е. сначала выполняется тело цикла, а потом идет сравнение.

```
do{ выполняются действия;
} while (условие);
```

- **for** - оператор цикла.

```
for (int a=1; a< 3; a++) // инициализируется переменная и ее начальное значение; условие выполнения
                         цикла; действие над переменной после выполнения тела цикла
{
выполняются действия; //команда, которая будет выполнятся пока в условие выполнения true
}
```

- **for-each** - это оператор для перебора массива/коллекций, разновидность оператора for.

```
int[] array = {1, 2, 3}; //создаем массив
for (int i : array) //циклический оператор for, внутри скобок указывается: тип и название
                    //(int i) : название массива/коллекции (array))
{
System.out.println(i); //команда для исполнения.
        }
```

21. Объясните работу оператора instanceof. Что будет результатом работы оператора, если слева от него будет стоять ссылка, равная null?

Оператор instanceof сравнивает объект с указанным типом. Можно использовать его для проверки, является ли объект экземпляром класса, экземпляром подкласса или экземпляром класса, который реализует определенный интерфейс. Возвращает значение boolean.

```
String str = “Hello”;
System.out.println(str instanceof String); //result = true
```

Применение instanceof в нулевой ссылочной переменной возвращает false.

```
String s = null;
System.out.println(s instanceof String); //result = false
```