Вопросы и ответы на собеседование по основам Java

(первая часть)

1). Что такое статические блоки и статическая инициализация в Java?

Статические блоки или статическая инициализация используются для инициализации статических полей в Java. Мы объявляем статические блоки, когда нам нужно инициализировать статические поля в нашем классе. Статические блоки выполняются ровно один раз, когда класс загружается. Статические блоки выполняются даже до вызова конструкторов.

2). Как вызвать один конструктор из другого конструктора?

Внутри того же класса, если мы хотим вызвать один конструктор из другого, мы используем метод **this()**. В зависимости от количества параметров, которые мы передаем, вызывается соответствующий метод this().

Ограничения при использовании этого метода:

1) this должен быть первым оператором в конструкторе.

2) Мы не можем использовать два метода this() в одном конструкторе.

3). Что такое переопределение метода (method overriding) в Java?

Если у нас есть методы с той же сигнатурой (тем же именем, тем же типом возвращаемого значения) в суперклассе и подклассе, то мы говорим, что подкласс переопределяет метод суперкласса.

Когда использовать переопределение в Java?

Если нам нужен тот же метод с разным поведением в суперклассе и подклассе, мы используем переопределение.

При вызове переопределенного метода ссылаясь на подкласс, вызывается метод подкласса, скрывая метод суперкласса.

4). Что такое ключевое слово super в Java?

Переменные и методы суперкласса могут быть переопределены в подклассе. В случае переопределения объект подкласса вызывает свои собственные переменные и методы. Подкласс не может получить доступ к переменным и методам суперкласса, потому что переопределенные переменные или методы скрывают методы и переменные суперкласса. Тем не менее, Java предоставляет способ доступа к членам суперкласса даже в случае их переопределения. Ключевое слово super используется для доступа к переменным, методам и конструкторам суперкласса.

Super может использоваться в двух формах:

1) Первая форма используется для вызова конструктора суперкласса.

2) Вторая форма используется для вызова переменных и методов суперкласса. Super, если присутствует, должен быть первым оператором.

5) В чем разница между перегрузкой методов (method overloading) и переопределением методов (method overriding) в Java?

METHOD OVERLOADING	METHOD OVERRIDING
1) Перегрузка методов происходит внутри одного класса.	1) Переопределение методов происходит между двумя классами: суперклассом и подклассом.
2) Так как при перегрузке методов участвует только один класс, наследование не участвует.	2) Поскольку переопределение методов происходит между суперклассом и подклассом, в этом участвует наследование.
3) При перегрузке методов тип возвращаемого значения не обязан быть одинаковым.	3) При переопределении методов тип возвращаемого значения должен быть одинаковым.
4) При перегрузке методов параметры(список аргументов) должны быть разными.	4) Параметры (список аргументов) в переопределенном методе должны быть теми же, что и в исходном методе.
5) Статический полиморфизм можно достичь с помощью перегрузки методов.	5) Динамический полиморфизм можно достичь с помощью переопределения методов.
6) При перегрузке один метод не может скрыть другой метод	6) При переопределении метода метод подкласса скрывает метод суперкласса.

6) В чем разница между абстрактным классом и интерфейсом?

INTERFACE	ABSTRACT CLASS
1) Интерфейс содержит только абстрактные методы.	1) Абстрактный класс может содержать абстрактные методы, конкретные методы или и то, и другое.
2) Модификаторы доступа для методов в интерфейсе по умолчанию считаются public.	2) Кроме private, для методов в абстрактном классе можно использовать любой модификатор доступа. Это связано с тем, что абстрактные классы служат в основном для создания общей структуры и поведения для подклассов.
3) Переменные в интерфейсе всегда считаются public , static , final.	3) Кроме переменных с модификатором private, переменные могут иметь любые модификаторы доступа.
4) Множественное наследование в Java реализуется с использованием интерфейсов. Классы могут реализовывать несколько интерфейсов, что позволяет достичь множественного наследования.	4) Множественное наследование не может быть достигнуто с использованием абстрактных классов, поскольку классы могут наследовать только один абстрактный класс.
5) Для реализации интерфейса в Java используется ключевое слово implements.	5) Для наследования (расширения) абстрактного класса в Java используется ключевое слово extends.

7). Почему Java является платформонезависимой?

Самая уникальная особенность Java - это платформонезависимость. В любом языке программирования исходный код компилируется в исполняемый код, который нельзя запустить на всех платформах. Когда **javac** компилирует программу на Java, он создает исполняемый файл с расширением .**class**. Файл .class содержит байт-коды. Байт-коды интерпретируются только **JVM** (Java Virtual Machine). Поскольку JVM доступна на всех платформах благодаря Sun Microsystems, мы можем выполнять этот байт-код на любой платформе. Байт-код, созданный в среде Windows, также может выполняться в среде Linux. Это делает Java **платформонезависимой**.

8). Что такое перегрузка методов в Java?

Класс, имеющий два или более метода с одинаковым именем, но с разными аргументами, считается перегруженным. Статический полиморфизм достигается в Java с помощью перегрузки методов. Перегрузка методов используется, когда мы хотим, чтобы методы выполняли похожие задачи, но с разными входными данными или значениями. Когда вызывается перегруженный метод, Java сначала проверяет имя метода и количество аргументов, тип аргументов, на основе чего компилятор выполняет этот метод. Компилятор решает, какой метод вызвать на этапе компиляции. С помощью перегрузки можно достичь статического полиморфизма или статической привязки в Java.
Примечание: Тип возвращаемого значения не является частью сигнатуры метода. Мы можем иметь методы с разными типами возвращаемого значения, но только тип возвращаемого значения недостаточен для вызова метода в Java.

9) В чем разница между C++ и Java?

JAVA	C++
1) Java является платформонезависимым.	1) C++ зависит от платформы (платформозависимый).
2) В Java отсутствуют указатели.	2) В C++ есть указатели.
3) В Java отсутствует перегрузка операторов.	3) C++ поддерживает перегрузку операторов.
4) В Java есть сборка мусора.	4) В C++ сборки мусора нет.
5) Java поддерживает многозадачность (многопоточность).	5) C++ не поддерживает многозадачность (многопоточность).
6) В Java отсутствуют шаблоны (templates).	6) В C++ есть шаблоны.
7) В Java отсутствуют глобальные переменные.	7) В C++ можно использовать глобальные переменные.

10). Что такое компилятор JIT в Java?

Компилятор **JIT** означает "Just in time compiler" (Компилятор "Только в нужный момент"). Компилятор JIT компилирует байт-код в исполняемый код. JIT является частью **JVM** (Java Virtual Machine). JIT не может преобразовать весь java-код в исполняемый код, он компилирует его только тогда, когда это необходимо во время выполнения.

11). Что такое байт-код в Java?

Когда компилятор **javac** компилирует класс, он создает файл с расширением **.class**. Этот файл .class содержит набор инструкций, называемых байт-кодом. Байт-код - это язык, независимый от конкретной аппаратной платформы, и содержит набор инструкций, которые должны выполняться только JVM (Java Virtual Machine). JVM может понимать эти байт-коды.

12). В чем разница между this() и super() в Java?

this() используется для вызова одного конструктора из другого внутри того же класса, в то время как **super()** используется для вызова конструктора суперкласса. Или this() или super(), если они присутствуют, должны быть **первым** оператором в конструкторе.

13). Что такое класс в Java?

Классы являются фундаментальными или основными единицами в объектно-ориентированном программировании. Класс представляет собой своего рода чертеж или шаблон для объектов. Класс определяет переменные и методы. Класс говорит, какого **типа** объекты мы создаем. Например, класс Department говорит нам, что мы можем создавать объекты типа Department. Мы можем создать любое количество объектов класса Department. Все программные конструкции в Java находятся внутри классов. Когда JVM начинает выполнение, она сначала ищет класс при компиляции. Каждое приложение на Java должно иметь как минимум один класс и один метод main. Класс начинается с ключевого слова **class**. Определение класса должно быть сохранено в файле класса с тем же именем, что и имя класса. Имя файла должно заканчиваться расширением **.java**.

```
public class FirstClass {
public static void main(String[] args)
{
System.out.println(“My First class”);
}
}
```

Если рассматривать FirstClass класс, то при компиляции JVM загружает класс FirstClass и создает файл **.class** (FirstClass.class). Когда мы запускаем программу, мы запускаем этот класс, а затем выполняем метод main.

14). Что такое объект в Java?

Объект является экземпляром класса. Класс определяет тип объекта. Каждый объект принадлежит какому-то классу. Каждый объект содержит состояние и поведение. Состояние определяется значением атрибутов, а поведение - методами. Объекты также называются экземплярами.
Для создания экземпляра класса мы объявляем переменную с типом класса.

```
public classFirstClass {
public static void main(String[] args)
{
FirstClass f = new FirstClass();
System.out.println(“My First class”);
}
}
```

Для создания экземпляра класса FirstClass мы используем следующее выражение:
FirstClass f = new FirstClass();
Здесь f используется для ссылки на объект класса FirstClass.

15). Что такое метод в Java?

Метод в Java - это блок кода, который может выполняться над конкретным объектом класса. Метод включает в себя имя метода, параметры или аргументы, тип возвращаемого значения и тело выполнимого кода.

Syntax :
type methodName(Argument List){
}

```
public float add(int a, int b, int c) {
    // Тело метода
}
```

Методы могут иметь несколько аргументов, и их названия перечисляются через запятую, если у нас есть несколько аргументов.

16). Что такое инкапсуляция в Java?

Инкапсуляция - это процесс упаковки или объединения данных в одну единицу (класс), и сохранение данных в безопасности от неправильного использования.
Через инкапсуляцию мы можем скрыть и защитить данные, хранящиеся в объектах Java. Java поддерживает инкапсуляцию с помощью уровней доступа. В Java существует четыре модификатора доступа: public, private, protected и уровень доступа по умолчанию.
Для примера, возьмем класс "автомобиль". Внутри автомобиля много деталей, которые водителю не обязательно знать. Водитель должен знать только, как завести и остановить машину. Мы можем выставить только те детали, которые требуются, и скрыть остальные, используя инкапсуляцию.

17). Почему метод main() является public, static и void в Java?

1. public: "public" - это модификатор доступа, который позволяет использовать метод за пределами класса. Когда метод main объявлен как public, это означает, что его можно использовать вне класса.
2. static: Для вызова метода требуется объект. Иногда может потребоваться вызвать метод без создания объекта. В этом случае метод объявляется как static. JVM вызывает метод main() без создания объекта, объявляя его с ключевым словом static.
3. void: Тип возвращаемого значения "void" используется, когда метод не возвращает какое-либо значение. Метод main() не возвращает значения, поэтому он объявлен как void.

Signature :

```
public static void main(String[] args) {
}
```

18). Расскажите о методе main() в Java?

Метод main() является точкой старта выполнения для всех Java-приложений.

public static void main(String[] args) {}

String args[] представляет собой массив строковых объектов, которые мы можем передавать как аргументы командной строки. Каждое Java-приложение должно иметь как минимум **один** метод main().

Signature :
public static void main(String[] args) {
}

19). Что такое конструктор в Java?

Конструктор - это специальный метод, используемый для инициализации объектов в Java. Мы используем конструкторы для инициализации всех переменных в классе при создании объекта. Как только объект создан, он автоматически инициализируется с помощью конструктора в Java. У нас есть два типа конструкторов:

1. Конструктор по умолчанию (Default Constructor)
2. Параметризированный конструктор (Parameterized Constructor)

```
public имяКласса() {}
```

```
public имяКласса(список Параметров) {}
```

20). В чем разница между методом length() и length в Java?

- **length()**: В классе String есть метод length(), который используется для возврата количества символов в строке.

Пример:
String str = "Привет, мир"; System.out.println(str.length());
str.length() вернет 11 символов, включая пробелы.

- **length**: У массивов есть переменная length, которая возвращает количество значений или объектов в массиве.

Пример:
String[] days={” Sun”,”Mon”,”wed”,”thu”,”fri”,”sat”};
days.length вернет 6, так как количество значений в массиве дней равно 6.

21). Что такое ASCII-код?

ASCII означает **American Standard Code for Information Interchange** (Американский стандартный код для обмена информацией). Диапазон символов ASCII составляет от 0 до 255. Мы не можем добавлять больше символов в набор символов ASCII. Набор символов ASCII поддерживает только английский язык. Поэтому, если мы говорим о языке C, то он написан только на английском языке, и его нельзя написать на других языках, потому что он использует код ASCII.

22). Что такое Unicode?

Unicode - это набор символов, разработанный Unicode Consortium. Чтобы поддерживать все языки мира, Java использует значения Unicode. Символы Unicode представлены 16-битными и их диапазон символов составляет от 0 до 65,535. В Java используется код ASCII для всех элементов ввода, кроме строк, идентификаторов и комментариев.

23). В чем разница между символьной константой и строковой константой в Java?

Символьная константа заключается в одинарные кавычки. Строковые константы заключены в двойные кавычки. Символьные константы представляют собой одну цифру или символ. Строковые константы - это собрание символов.

Пример символьной константы: '2', 'A'

Пример строковой константы: "Hello World"

24). Что такое константы и как создавать константы в Java?

Константы - это фиксированные значения, значения которые не могут быть изменены во время выполнения программы. В Java мы создаем константы с помощью ключевого слова **final**.

Пример:

```
final int number = 10;
final String str = "java-interview-questions";
```

25). В чем разница между операторами '>>' и '>>>' в Java?

- '>>' - это оператор сдвига вправо, который сдвигает все биты значения вправо на указанное количество раз.

Например:

int a = 15;

a = a >> 3;

Этот код сдвигает число 15 на три позиции вправо.

- '>>>' - это оператор сдвига вправо без знака, используется для сдвига вправо. Позиции, освободившиеся после сдвига, заполняются нулями.

26). Объясните стандарты кодирования Java для классов или соглашения о кодировании Java для классов?

Sun создала стандарты кодирования Java или соглашения о кодировании Java. Настоятельно рекомендуется следовать стандартам кодирования Java.

- Имена классов должны начинаться с заглавной буквы.
- Имена классов должны быть существительными. Если имя класса состоит из нескольких слов, то первая буква каждого следующего слова должна быть заглавной.

Примеры: Employee, EmployeeDetails, ArrayList, TreeSet, HashSet

27). Объясните стандарты кодирования Java для интерфейсов?

- Имена интерфейсов должны начинаться с заглавных букв.
- Имена интерфейсов должны быть прилагательными.

Примеры: Runnable, Serializable, Marker, Cloneable

28). Объясните стандарты кодирования Java для методов?

- Имена методов должны начинаться с маленьких букв.
- Имена методов, как правило, являются глаголами.
- Если имя метода состоит из нескольких слов, то каждое следующее слово должно начинаться с заглавной буквы.
  Пример: toString()
- Имя метода должно быть комбинацией глагола и существительного.
  Примеры: getCarName(), getCarNumber()

29). Объясните стандарты кодирования Java для переменных?

- Имена переменных должны начинаться с маленьких букв.
- Имена переменных должны быть существительными.
- Рекомендуется использовать краткие и значимые имена.
- Если в имени переменной есть несколько слов, то каждое внутреннее слово должно начинаться с заглавной буквы. Примеры: string, value, empName, empSalary

30). Объясните стандарты кодирования Java для констант?

Константы в Java создаются с использованием ключевых слов static и final.

- Имена констант содержат только заглавные буквы.
- Если имя константы состоит из двух слов, они должны разделяться подчеркиванием.
- Имена констант, как правило, являются существительными.

Примеры: MAX_VALUE, MIN_VALUE, MAX_PRIORITY, MIN_PRIORITY