Вопросы для подготовки Java Core + ООП:

- На какие основные группы можно поделить типы данных в Java?

Примитивные и ссылочные

- Какие примитивные типы вы знаете?

Byte, short, int, long, float, double, Boolean, char

- Что вы знаете о преобразовании примитивных типов данных, есть ли потеря данных?

Int -> float, long -> float, long -> double с потерей точности (автоматические)

При явном преобразовании и downcaste будет целочисленное переполнение и

Усечение до n первых разрядов.

Из дробных чисел в целые – откидывается дробная часть

- Какими значениями инициализируются переменные по умолчанию?

0, 0.0, false

- Как передается значение переменной (по ссылке/значению)?

Примитивные типы – значение, ссылочные - ссылка

- Что такое тернарный оператор выбора?

Оператор, принимающий три операнда.

Синтаксис: <выражение>?<выражение если истинно>?<выражение если ложно>

- Какие унарные и бинарные арифметические операции вы знаете?

Унарные (воздействуют только на одно значение или выражение): - (унарный минус, меняет знак числа), + (унарный плюс, не выполняет никаких действий), ~ (побитовое дополнение, инвертирует все биты ЦЕЛОГО числа, 0 меняет на 1, 1 на 0), ++ (инкремент для целых), -- (декремент для целых)

Бинарные: + (сложение), - (вычитание), \* (умножение), / (деление), % (остаток)

- Какие побитовые операции вы знаете?

& - побитовое И (\*)  
 | - побитовое ИЛИ (+)

^ - исключающее ИЛИ (если биты не равны, то 1, иначе 0)

<< - побитовый сдвиг влево с учетом знака

>> - побитовый сдвиг вправо с учетом знака

>>> - побитовый сдвиг вправо без учета знака

Сдвиг: x = 30; z = 30 << 2 (2 – на сколько битов сдвинуть)

- Какова роль и правила написания оператора выбора (switch)?

Switch позволяет избежать множества if, else-if, else-if…

switch (expression) {

case(value1):

// code

break;

case(value2):

//code

break;

default:

//code

break;

}

Без break код продолжит выполнение до следующего break или до конца switch

- Какие циклы вы знаете, в чем их отличия?

for – обычно используется для прохода по элементам массива С ВОЗМОЖНОСТЬЮ ИМЕТЬ ДОСТУП К ОСТАЛЬНЫМ ЭЛЕМЕНТАМ МАССИВА ПО ИНДЕКСУ

foreach – обычно используется для прохода по элементам массива и их вывода (БЕЗ ДОСТУПА К ОСТАЛЬНЫМ ЭЛЕМЕНТАМ МАССИВА)

while – выполняется пока истинно условие в скобочках

do-while – выполняется пока истинно условие в скобочках, но обязательно выполнится хотя бы 1 раз, даже если условие изначально ложно.

- Какие параметры имеет цикл for, можно ли их не задать?

for (<инициализация>;<условие выхода из цикла>;<действие после каждой итерации>) {}

можно не задать чтобы сделать бесконечный цикл: for(;;) {}b

- Какой оператор используется для немедленной остановки цикла?

break

- Какой оператор используется для перехода к следующей итерации цикла?

continue

- Что такое массив?

Структура данных, которая может хранить множество элементов ОДНОГО ТИПА

- Что вы знаете о классах оболочках?

Они есть для каждого примитивного типа данных, только начинаются с заглавной буквы. (double -> Double, long -> Long, но char -> Character, int -> Integer).

Объекты классов оболочек неизменяемые (Immutable), все имеют переопределённый метод equals. Все наследуют абстрактный класс Number, содержат static константы (Integer.MAX\_VALUE). Обёртки Float и Double так же содержат отрицательную бесконечность, положительную бесконечность и NaN (Not A Number), все имеют переопределённый метод toString(). Некоторые коллекции работают только с объектами, поэтому нужны оболочки. Так же они нивелируют недостатки примитивных типов – примитивы не имеют методов.

- Что такое автоупаковка (boxing/unboxing)?

Автоупаковка и автораспаковка одна из особенностей примитивов и их классов-обёрток. Java – ООП язык, но примитивы не являются объектами, но при этом классу-обёртке можно присвоить значение примитивного типа – это называется АВТОУПАКОВКА. А когда мы примитивному типу присваиваем объект класса обёртки – это называется АВТОРАСПАКОВКА.

Параметры методов тоже подлежат автоупаковке и автораспаковке, но для массивов автораспаковка и автоупаковка не работают.

- Назовите принципы ООП и расскажите о каждом.

1. Наследование

2. Абстракция

3. Инкапсуляция (сокрытие)

4. Полиморфизм

1)Наследование: механизм, который позволяет описать новый класс на основе существующего (родительского). При этом свойства и функциональность родительского класса заимствуются новым классом. Позволяет повторно использовать код т.к поля и методы, описанные в родительских классах можно использовать в классах-наследниках.

2) Абстракция означает выделение главных, наиболее значимых характеристик предмета и наоборот — отбрасывание второстепенных, незначительных.

3) Инкапсуляция (сокрытие): означает ограничение доступа к данным и возможностям их изменения. В Java обеспечивается механизм сокрытия, позволяющий разграничивать доступ к различным частям компонента. (в основе лежит слово «капсула». В эту «капсулу» мы прячем какие-то важные для нас данные, которые не хотим, чтобы кто-то менял)

4) Полиморфизм: даёт возможность использовать одни и те же методы для объектов разных классов, при этом поведение объектов будет разным в зависимости от типа, к которому они принадлежат.

- Дайте определение понятию “класс”.

Класс это шаблон/описание объекта.

- Что такое поле/атрибут класса?

Поле класса — это переменная, которая описывает какое-либо из свойств данного класса.

- Как правильно организовать доступ к полям класса?

Сделать поля класса приватными, а геттеры и сеттеры - публичными

- Дайте определение понятию “конструктор”.

Конструкторы - это специальные методы, которые вызываются при создании объекта. Они создают новый объект определенного класса.

- Чем отличаются конструктор по-умолчанию и конструктор с параметрами?

Конструктор по умолчанию (default constructor) – это конструктор, который не имеет параметров. Конструктор по умолчанию может объявляться в классе явным образом или генерироваться автоматически. Конструкторы с параметрами в качестве параметров принимают данные и создают новый объект с ними.

- Какие модификаторы доступа вы знаете, расскажите про каждый из них.

private — доступен только методам этого класса

public — доступен методам любых классов

protected — доступен методам этого класса и методам наследников

У интерфейса все методы только public.

По умолчанию поле и метод имеют модификатор доступа package-private, отличный от всех трех. Обсуждение свойств этого модификатора — в отдельной теме про package и его использование.

- О чем говорят ключевые слова “this”, “super”, где и как их можно использовать?

this и super - это два специальных ключевых слова в Java, которые представляют соответственно текущий экземпляр класса и его суперкласса.

главное отличие между this и super в Java в том, что this представляет текущий экземпляр класса, в то время как super - текущий экземпляр родительского класса.

- Дайте определение понятию “метод”.

То, что умеет делать класс (функции класса)

- Что такое сигнатура метода?

Сигнатура метода — это имя метода плюс параметры (причем порядок параметров имеет значение). В сигнатуру метода не входит возвращаемое значение, а также бросаемые им исключения.

- Какие методы называются перегруженными?

Перегрузка методов — это приём программирования, который позволяет разработчику в одном классе для методов с разными параметрами использовать одно и то же имя. В этом случае мы говорим, что метод перегружен.

- Могут ли нестатические методы перегрузить статические?

Да. В итоге получится два разных метода. Статический будет принадлежать классу и будет доступен через его имя, а нестатический будет принадлежать конкретному объекту и доступен через вызов метода этого объекта.

- Расскажите про переопределение методов.

Переопределение метода в ООП - одна из возможностей языка программирования, позволяющая подклассу или дочернему классу обеспечивать специфическую реализацию метода, уже реализованного в одном из суперклассов или родительских классов.

- Может ли метод принимать разное количество параметров (аргументы

переменной длины)?

Да, это будет метод с аргументами переменной длинны.

- Чем отличается переопределение от перегрузки?

При перегрузке вы создаёте внутри одного класса много методов с одинаковым названием, но разными параметрами. При переопределении вы берёте один и тот же метод и заставляете его делать разные вещи в зависимости от того, в каком классе его вызвали

- Зачем нужен оператор instanceof?

instanceof используется для определения типа объекта во время выполнения программы. Определив тип объекта можно реализовать корректное приведение типов в случае, когда классы образуют иерархию.

- Зачем нужны и какие бывают блоки инициализации?

Блоки инициализации представляют собой наборы выражений инициализации полей, заключенные в фигурные скобки и размещаемые внутри класса вне объявлений методов или конструкторов. Блок инициализации выполняется так же, как если бы он был расположен в верхней части тела любого конструктора. Если блоков инициализации несколько, они выполняются в порядке следования в тексте класса. Блок инициализации способен генерировать исключения, если их объявления перечислены в предложениях throws всех конструкторов класса.

Бывают статические и нестатические блоки инициализации. Также возможно создать такой блок в анонимном классе.

- Каков порядок вызова конструкторов и блоков инициализации двух классов: потомка и его предка?

Сначала вызываются все статические блоки от первого предка до последнего наследника. Потом попарно вызываются динамический блок инициализации и конструктор в той же последовательности (от предка до последнего потомка).

- Где и для чего используется модификатор abstract?

Нельзя создать методы класса с модификатором abstract (абстрактный), но при этом наследники класса могут быть не абстрактными.

Так же есть абстрактные методы – методы, которые не имеют реализации. Если есть хоть один такой метод в классе – то класс должен быть абстрактным.

- Можно ли объявить метод абстрактным и статическим одновременно?

Нет. Эти модификаторы нельзя совмещать.

- Что означает ключевое слово static?

Static говорит о том, что метод/поле класса принадлежат не объекту, а классу, значит мы можем использовать статические методы и поля не создавая экземпляр класса. Но такие методы могут использовать только другие static методы и использовать только static переменные, не могут использовать this и super т.к это не объекты.

- К каким конструкциям Java применим модификатор static?

Метод, класс, поле, внутренний класс.

- Можно ли перегрузить static метод?

Static метод можно перегрузить, но не переопределить.

- Что такое статический класс, какие особенности его использования?

Статический класс - вложенный класс, который может обращаться только к статическим полям обертывающего его класса, в том числе и приватным. Доступ к нестатическим полям обрамляющего класса может быть осуществлен только через ссылку на экземпляр обрамляющего объекта. К классу высшего уровня модификатор static неприменим

- Как влияет модификатор static на класс/метод/поле?

Static говорит о том, что метод/поле класса принадлежат не объекту, а классу, значит мы можем использовать статические методы и поля не создавая экземпляр класса

- О чем говорит ключевое слово final?

final класс – не может иметь наследков

final метод – не может быть переопределен

final переменная – не может изменить значение после инициализации

- Дайте определение понятию “интерфейс”.

Interface – полностью абстрактный класс, интерфейс определяет методы, их имена, список аргументов и тип возвращаемого значения которые должен иметь класс, который его реализует.

- Какие модификаторы по умолчанию имеют поля и методы интерфейсов?

public static final

- Какие особенности создания вложенных классов: простых и статических.

Внутренние классы - это классы, объявленные внутри другого класса без использования модификатора static. Такие классы имеют доступ к членам внешнего класса, включая приватные поля и методы, и могут использоваться для создания более читаемого и логически законченного кода.

Статические вложенные классы - это классы, объявленные внутри другого класса с использованием модификатора static. Эти классы не имеют доступа к членам внешнего класса и используются для логической группировки классов и для создания пространства имен.

- Что вы знаете о вложенных классах, зачем они используются? Классификация, варианты использования, о нарушении инкапсуляции.

Вложенный класс — это класс, который объявлен внутри объявления другого класса. Вложенные классы делятся на статические и нестатические (non-static). Собственно нестатические вложенные классы имеют и другое название — внутренние классы (inner classes).

Внутренние классы в Java делятся на такие три вида:

внутренние классы-члены (member inner classes);

локальные классы (local classes);

анонимные классы (anonymous classes).

- В чем разница вложенных и внутренних классов?

Внутренние классы-члены ассоциируются не с самим внешним классом, а с его экземпляром. При этом они имеют доступ ко всем его полям и методам.

Локальные классы (local classes) определяются в блоке Java кода. На практике чаще всего объявление происходит в методе некоторого другого класса. Хотя объявлять локальный класс можно внутри статических и нестатических блоков инициализации.

Использование вложенных классов всегда приводит к некоторому нарушению инкапсуляции — вложенный класс может обращаться к закрытым членам внешнего класса (но не наоборот!).

- Какие классы называются анонимными?

Анонимный класс - это локальный класс без имени.

- Каким образом из вложенного класса получить доступ к полю внешнего класса?

Если вложенный класс не статический, то можно просто обратиться к этому полю из внутреннего класса.

- Каким образом можно обратиться к локальной переменной метода из анонимного класса, объявленного в теле этого метода? Есть ли какие-нибудь ограничения для такой переменной?

Анонимный класс имеет доступ к полям внешнего класса.

Анонимный класс не имеет доступ к локальным переменным области, в которой он определен, если они не финальные (final) или неизменяемые.

Как и у других внутренних классов, объявление переменной с именем, которое уже занято, затеняет предыдущее объявление.

Вы не можете определять статические члены анонимного класса.

- Как связан любой пользовательский класс с классом Object?

ВСЕ классы неявно наследуются от суперкласса Object.

- Расскажите про каждый из методов класса Object.

getClass() — возвращает в рантайме класс данного объекта.

int hashCode() — возвращает хеш-код

boolean equals(Object obj) — сравнивает объекты по значениям.

Object clone() — клонирование объекта

String toString() — возвращает строковое представление объекта.

void notify() — просыпается один поток, который ждет на “мониторе” данного объекта.

void notifyAll() — просыпаются все потоки, которые ждут на “мониторе” данного объекта.

void wait(long timeout) — поток переходит в режим ожидания в течение указанного времени.

void wait() — приводит данный поток в ожидание, пока другой поток не вызовет notify() или notifyAll() методы для этого объекта.

void wait(long timeout, int nanos) — приводит данный поток в ожидание, пока другой поток не вызовет notify() или notifyAll() для этого метода, или пока не истечет указанный промежуток времени.

void finalize() — вызывается сборщиком мусора, когда garbage collector определил, что ссылок на объект больше нет.

- Что такое метод equals(). Чем он отличается от операции ==.

== сравнивает по ссылкам, equals сравнивает объекты в зависимости от реализации этого метода (обычно по значениям).

- Если вы хотите переопределить equals(), какие условия должны удовлетворяться для переопределенного метода?

Рефлексивность: для любого ненулевого x, x.equals(x) вернет true;

Транзитивность: для любого ненулевого x, y и z, если x.equals(y) и y.equals(z) вернет true, тогда и x.equals(z) вернет true;

Симметричность: для любого ненулевого x и y, x.equals(y) должно вернуть true, тогда и только тогда, когда y.equals(x) вернет true.

Также для любого ненулевого x, x.equals(null) должно вернуть false

- Если equals() переопределен, есть ли какие-либо другие методы, которые следует переопределить?

Переопределил equals – переопредели и hashCode. Одинаковые объекты должны иметь одинаковые хэшкоды.

- В чем особенность работы методов hashCode и equals? Каким образом реализованы методы hashCode и equals в классе Object? Какие правила и соглашения существуют для реализации этих методов? Когда они применяются?

Если хеш-коды разные, то и входные объекты гарантированно разные.

Если хеш-коды равны, то входные объекты не всегда равны.

- Какой метод возвращает строковое представление объекта?

toString()

- Что будет, если переопределить equals не переопределяя hashCode? Какие могут возникнуть проблемы?

Классы и методы, которые использовали эти методы могут некорректно работать. Нарушится работа hashmap

- Есть ли какие-либо рекомендации о том, какие поля следует использовать при подсчете hashCode?

Все поля, и чем «сложнее» функция которая считает хэшкод – тем лучше

- Как вы думаете, будут ли какие-то проблемы, если у объекта, который используется в качестве ключа в hashMap изменится поле, которое участвует в определении hashCode?

При обращении по ключу можно не найти значение.

- Чем отличается абстрактный класс от интерфейса, в каких случаях что вы будете использовать?

Интерфейс описывает только поведение. У него нет состояния. А у абстрактного класса состояние есть: он описывает и то, и другое.

Абстрактный класс связывает между собой и объединяет классы, имеющие очень близкую связь. В то же время, один и тот же интерфейс могут реализовать классы, у которых вообще нет ничего общего.

Классы могут реализовывать сколько угодно интерфейсов, но наследоваться можно только от одного класса.

В итоге с абстрактными классами вы теряете индивидуальность класса, наследующего его; с интерфейсами вы просто расширяете функциональность каждого класса.

- Можно ли получить доступ к private переменным класса и если да, то каким образом?

Внутри класса или через getter/setter

- Что такое volatile и transient? Для чего и в каких случаях можно было бы использовать default?

volatile — этот модификатор вынуждает потоки отключить оптимизацию доступа и использовать единственный экземпляр переменной. Если переменная примитивного типа – этого будет достаточно для обеспечения потокобезопасности. Если же переменная является ссылкой на объект – синхронизировано будет исключительно значение этой ссылки. Все же данные, содержащиеся в объекте, синхронизированы не будут!

transient — указание того, что при сериализации/десериализации данное поле не нужно сериализовать/десериализовывать.

- Имеет ли смысл объявлять метод private final?

Нет, такой метод и так не видят наследники, а значит его не могут переопределить.

- Какие особенности инициализации final переменных?

final поле не может менять значение после инициализации, не статическое можно инициализировать в статическом и динамическом блоке инициализации.

- Что будет, если единственный конструктор класса объявлен как final?

Конструктор нельзя сделать final

- Что такое finalize? Зачем он нужен? Что Вы можете рассказать о сборщике мусора и алгоритмах его работы.

Finalize() вызывается перед тем, как объект будет удален сборщиком мусора. Есть много разных реализаций gc. Основа работы следующая: gc помечает объекты, на которые больше не ссылаются другие объекты для их удаления (считает ссылки). Затем на одном из проходов помеченные объекты удаляются.

Вызов finalize не гарантирован, так как сборщик мусора может не успеть выполнить проход до закрытия приложения.

- Почему метод clone объявлен как protected? Что необходимо для реализации клонирования?

Метод clone нужно переопределить, для этого нужно реализовать интерфейс Cloneable

- Можно ли наследовать строковый тип, почему?

Нельзя, все строковые классы - final

- Дайте определение понятию конкатенация строк.

Это объединение/сложение/сцепление нескольких строк в одну.

- Как преобразовать строку в число?

С помощью методов классов-обёрток примитивных типов.(Integer.parseInt, Long.parseLong, etc…)

- Как сравнить значение двух строк?

С помощью equals()

- Как перевернуть строку?

Можно пройтись циклом for с конца до начала, записывая буквы в StringBuilder или использовать метод reverse объекта класса StringBuilder

- Как работает сравнение двух строк?

По equals сравниваются значения, по == сравниваются ссылки, т.к строка не примитивный тип.

- Чем отличаются и что общего у классов String, StringBuffer и StringBuilder?

String: immutable

StringBuilder: mutable, подходит для однопоточных приложений т.к методы не синхронизированы, но быстрее чем StringBuffer

StringBuffer: mutable, все методы синхронизированы, подходит для многопоточных