**1 Что такое autoboxing?**  
Автоупаковка это механизм неявной инициализации объектов классов-оберток (Byte, Short, Character, Integer, Long, Float, Double) значениями соответствующих им исходных примитивных типов (соотв. byte, short, char, int, long, float, double), без явного использования конструктора класса.  
Автоупаковка происходит при прямом присвоении примитива — классу-обертке (с помощью оператора"="), либо при передаче примитива в параметры метода (типа «класса-обертки»). Автоупаковке в «классы-обертки» могут быть подвергнуты как переменные примитивных типов, так и константы времени компиляции(литералы и final-примитивы). При этом литералы должны быть синтаксически корректными для инициализации переменной исходного примитивного типа.  
  
Автоупаковка переменных примитивных типов требует точного соответствия типа исходного примитива — типу «класса-обертки».  
Например, попытка автоупаковать переменную типа byte в Short, без предварительного явного приведения byte->short вызовет ошибку компиляции.  
  
Автоупаковка констант примитивных типов допускает более широкие границы соответствия. В этом случае компилятор способен предварительно осуществлять неявное расширение/сужение типа примитивов. Преобразование происходит в два этапа:  
1) неявное расширение(сужение) исходного типа примитива до типа примитива соответствующего классу-обертке (для преобразования int->Byte, сначала компилятор неявно сужает int в byte)  
2) автоупаковку примитива в соотвествующий «класс-обертку» (компилятор автоупаковывает byte->Byte). однако в этом случае существуют два дополнительных ограничения:  
a) присвоение примитива — «обертке» может производится только оператором "=" (нельзя передать такой примитив в параметры метода, без явного приведения типов)  
b) тип левого операнда не должен быть старше чем Character, тип правого не дожен старше чем int, (допустимо расширение/сужение byte <-> short, byte <-> char, short <-> char  
и только сужение byte < — int, short < — int, char < — int, все остальные варианты требуют явного приведения типов)  
  
Дополнительная особенность целочисленных «классов-оберток» созданных автоупаковкой констант в диапазоне -128 +127, в то что они кэшируются JVM. Потому такие обертки с одинаковыми значениями будут являтся ссылками на один объект.  
**2 Зачем используется autoboxing?**  
Я процитирую лекцию:  
Насколько ты помнишь, в Java есть как типы, унаследованные от класса Object, так и примитивные типы. Но, как оказалось, такая удобная вещь как коллекции и generic’и могут работать только с типами, унаследованными от Object.  
**3 Альтернативы autoboxing?**  
Не нашел ответа, но запостил на StackOverFlow.  
[stackoverflow.com/questions/39212744/what-is-the-alternative-to-autoboxing-in-java](http://stackoverflow.com/questions/39212744/what-is-the-alternative-to-autoboxing-in-java)  
Исходя из этой дискуссии, получается, что альтернатива autoboxing`у это использование примитивных типов, так как использовние autoboxing`a снижает производительность. Вывод: использовать autoboxing только там где это необходимо.  
Написана статья про Autoboxing:   
[effective-java.com/2010/05/the-advantages-and-traps-of-autoboxing/](https://effective-java.com/2010/05/the-advantages-and-traps-of-autoboxing/)  
**4 Типы-обертки для примитивных типов mutable или immutable?**  
Immutable, так как примитивные объекты тоже immutable. Чтобы работать как с Mutable типом есть класс MutableInteger, и.т.д.  
**5 Как примитивные типы приводятся к непримитивным аналогам?**  
На этот и последующий вопросы хорошо отвечает вот эта статья: [info.javarush.ru/translation/2013/08/22/%D0%90%D0%B2%D1%82%D0%BE%D1%83%D0%BF%D0%B0%D0%BA%D0%BE%D0%B2%D0%BA%D0%B0-%D0%B8-%D1%80%D0%B0%D1%81%D0%BF%D0%B0%D0%BA%D0%BE%D0%B2%D0%BA%D0%B0-%D0%B2-Java-.html](http://info.javarush.ru/translation/2013/08/22/%D0%90%D0%B2%D1%82%D0%BE%D1%83%D0%BF%D0%B0%D0%BA%D0%BE%D0%B2%D0%BA%D0%B0-%D0%B8-%D1%80%D0%B0%D1%81%D0%BF%D0%B0%D0%BA%D0%BE%D0%B2%D0%BA%D0%B0-%D0%B2-Java-.html)  
Это заключение из нее:  
Автоупаковка является механизмом для скрытого преобразования примитивных типов данных в соответствующие классы-оболочки(объекты). Компилятор использует метод valueOf() чтобы преобразовать примитивные типы в объекты, а методы IntValue (), doubleValue () и т.д., чтобы получить примитивные типы из объекта(То есть обратное преобразование). Автоупаковка преобразует логический тип boolean в Boolean, byte в Byte, char в Character, float в Float, int в Integer, long в Long, short в Short. Распаковка происходит в обратном направлении.  
**6 Как непримитивные типы приводятся к примитивным?**  
Выше ответил  
**7 Как сравниваются примитивные и непримитивные типы?**  
В лекции это подробно рассматривается, но я нашел так скажем тоже самое, но другими словами.  
В Java есть два способа сравнивать объекты на равенство, == и метод equals().  
== используется для примитивных типов.  
Для объектов «==» это исключительно сравнение ссылок.  
Для остальных случаев нужно использовать метод equals(). Кроме того метод hashCode() служит (в теории) для той же цели. Хорошим тоном считается, если вы переопределили equals() и hashCode(). После инициализации неких объектов a и b должно выполняться правило:  
Если выражение a.equals(b) вернет true, то a.hashCode() должен быть равен b.hashCode().  
**8 Всегда ли создается новый объект при операции autoboxing?**  
Это в лекциях есть:  
  
Когда мы присваиваем переменной типа Integer значение типа int, при этом вызывается метод Integer.valueOf:  
функция valueOf не всегда создает новый объект типа Integer.   
Она кэширует значения от -128 до 127.  
Если передаваемое значение выходит за эти пределы, то новый объект создается, а если нет, то нет.  
Если мы пишем new Integer(), то гарантированно создается новый объект.  
Если мы вызываем Integer.valueOf(), явно или при autoboxing, то этот метод может вернуть для нас как новый объект, так и отдать объект из кэша, если переданное число лежит в диапазоне от -128 до 127.  
**9 Как работает кэширование при операции autoboxing?**  
Ответил в вопросе выше, на всякий случай создал вопрос на StackOverFlow, но там отвечают тоже самое:  
[stackoverflow.com/questions/39314971/how-cache-works-in-autoboxing-operation-java](http://stackoverflow.com/questions/39314971/how-cache-works-in-autoboxing-operation-java)  
**10 Для каких типов и/или значений работает кэширование?**  
В восьмом вопросе. Если у кого – то есть соображения на тему трех последних вопросов, то напишите в комментариях.