OOP, Classes and Objects, Inheritance, Class Object, Interfaces

1. Дайте развернутое объяснение трем концепциям ООП.

Инкапсуляция - это механизм языка, позволяющий объединить данные и методы, работающие с этими данными в единый объект и скрыть детали реализации от пользователя.

Наследование - это механизм включения методов и переменных базового класса в производный класс, таким образом он становиться доступен внутри дочернего класса.Наследование обеспечивает формальный механизм повторного использования кода.
Таким образом производный класс наследует все свойства родительского класса.

Полиморфизм - это механизм, позволяющий работать с разными типами данных так, словно это один и тот же тип. Простым языком, мы можем создавать одноименные методы (выполняющие одно и тоже логическое действие) для разных типов или наборов данных.

2. **Опишите процедуру инициализации полей класса и полей экземпляра класса. Когда инициализируются поля класса, а когда – поля экземпляров класса. Какие значения присваиваются полям по умолчанию? Где еще в классе полям могут быть присвоены начальные значения?**

Поля класса инициализируются в момент первой загрузки класса. Поля экземпляра класса инициализируются во время создания экземпляра (вызова new). По умолчанию ссылочным полям присваивается null, а примитивам - особые значения(0, 0.0 false)
Начальные значения можно присвоить в конструкторе или в статическом блоке.

3. **Приведите правила, которым должен следовать компонент java-bean**

1)Все поля должны быть private (используем get'еры и set'еры для доступа к ним)
2)public конструктор без аргументов
3)Имплементирует serializable интерфейс.

4. **Дайте определение перегрузке методов. Как вы думаете, чем удобна перегрузка методов? Укажите, какие методы могут перегружаться, и какими методами они могут быть перегружены? Можно ли перегрузить методы в базовом и производном классах? Можно ли private метод базового класса перегрузить public методов производного? Можно ли перегрузить конструкторы, и можно ли при перегрузке конструкторов менять атрибуты доступа у конструкторов?**

Перегрузка методов - это объявление методов с одним и тем же именем внутри класса, но с различными типами данных или их количеством. Перегрузка удобна тем, что для разных типов данных можно описать общее поведение. Перегрузка доступна как в базовом, так и в дочернем классе.

Private методы не видны за пределами класса, а значит и перегрузить их в дочернем классе нельзя.
Конструкторы можно также перегружать, а так же делать private.

5. **Объясните, что такое раннее и позднее связывание? Перегрузка – это раннее или позднее связывание? Объясните правила, которым следует компилятор при разрешении перегрузки; в том числе, если методы перегружаются примитивными типами, между которыми возможно неявное приведение или ссылочными типами, состоящими в иерархической связи.**

Раннее связывание - это когда метод, который будет вызван, известен во время компиляции.
Позднее связывание - это когда метод, который будет вызван, определяется во время компиляции.
Перегрузка - это раннее связывание, а переопределение - позднее связывание.
Если есть точное соответствие по типу и количеству параметров, то вызывается именно этот метод. Если нет, то компилятор в начале расширяет примитивные типы, а затем происходит автобоксинг или анбоксинг. И в конце идёт преобразование к Object. И в последнюю очередь идёт преобразование к методу с переменным числом аргументов того же типа, что и примитивная переменная.
Для ссылочных типов возможен преобразование дочерних типов к родительскому.

6. **Объясните, как вы понимаете, что такое неявная ссылка this? В каких методах эта ссылка присутствует, а в каких – нет, и почему?**

Каждому не статическому методу в качестве одного из параметров неявно передается ссылка this, указывающая на объект у которого был вызван данный метод. Начиная с JDK 8 ссылку можно явно передавать в качестве первого параметра метода. Это сделано для возможности аннотировать.
В статических методах нет ссылки this, т.к. эти методы принадлежат классу, а не объекту. К ним можно обратиться по имени класса.

7. **Что такое финальные поля, какие поля можно объявить со спецификатором final? Где можно инициализировать финальные поля?**

Финальные поля - это константы. final могут быть как поля класса, так и параметры метода и локальные переменные. Финальные поля можно инициализировать или сразу при объявлении, или в конструкторе.

8. **Что такое статические поля, статические финальные поля и статические методы. К чему имеют доступ статические методы? Можно ли перегрузить и переопределить статические методы? Наследуются ли статические методы?**

Поля, объявленные с модификатором static, являются общими для всех экземпляров класса. Они инициализируются при первой загрузке класса, как и static блоки. Static final объявляются константы. Статические методы, как и статические поля, доступны без обращения к конкретному экземпляру класса. Статические методы могут обращаться только к статическим переменным и методам. Статические методы можно перегружать, но нельзя переопределять. Фактически статические методы наследуются.

9. **Что такое логические и статические блоки инициализации? Сколько их может быть в классе, в каком порядке они могут быть размещены и в каком порядке вызываются?**