**Class Object, Inheritance, Interfaces**

1. **Чем является класс Object? Перечислите известные вам методы класса Object, укажите их назначение.**

Object – «основной» класс в Java. Каждый класс в jave неявно наследует Object (если не указано явное наследование). То есть методы класса Object можно вызывать на любом классе.

Методы:

*Equals* – сравнивает ссылки на объекты (необх. Переопределять)

*HashCode* - считать хэш объекта (необх. Переопределять)

*Wait*

*Notify/notifyAll*

*ToString* – выводит информацию о объекте в строковом формате (необх. Переопределять).

*GetClass* – возвращает имя класса объекта

1. **Что такое хэш-значение? Объясните, почему два разных объекта могут сгенерировать одинаковые хэш-коды?**

Хэш-значение – сгенерированный ключ (значение) для каждого объекта в Java программе. Используется сложная система подсчета, однако память Java машины ограниченна, поэтому теоретически разные объекты могут сгенерировать один хеш код, поэтому помимо хеш кода есть метод equals. То есть разные объекты могут иметь одинаковые хеш код, но одинаковые объекты не могут иметь разные хеш код.

1. **Как вы думаете, для чего используется наследование классов в java-программе? Приведите пример наследования. Как вы думаете, поля и методы, помеченными модификатором доступа private, наследуются?**

Наследование может использоваться для более понятного, структурированного кода и уменьшения его объема. Например если нужно создать множество разных классов, которые будут иметь какие-то общие свойства, можно объявит класс родитель и наследоваться от него.

Производный класс имеет доступ ко всем методам и полям базового класса (даже если базовый класс находится в другом пакете) кроме тех, которые определены с модификатором **private**. При этом производный класс также может добавлять свои поля и методы

1. **Укажите, как вызываются конструкторы при создании объекта производного класса? Что в конструкторе класса делает оператор super()? Возможно ли в одном конструкторе использовать операторы super() и this()?**

При создании производного класса вызываются конструкторы всех базовых классов. Оператор super() може явно передавать значение в конструктор базового класса. Оператор super. Может обращаться к полям базового класса. Super и this в одном к-ре использовать нельзя, т.к. оба должны быть на первой строчке к-ра.

1. **Объясните, как вы понимаете утверждения: “ссылка базового класса может ссылаться на объекты своих производных типов” и “объект производного класса может быть использован везде, где ожидается объект его базового типа”. Верно ли обратное и почему?**

Например если есть Class A и класс B extends A, то можно сделать:

A a = new B();

Точно также если есть List<A> list, то в него можно добавить и B переменную, то есть

B b = new B();

List.add(B); - все ок, скомпилируется.

Обратно это работать не будет.

1. **Что такое переопределение методов? Как вы думаете, зачем они нужны? Можно ли менять возвращаемый тип при переопределении методов? Можно ли менять атрибуты доступа при переопределении методов? Можно ли переопределить методы в рамках одного класса?**

Переопределить метод – значит изменить логику его действия, оставив имя метода тем же. Возвращаемое значение должно быть одно и то же. Атрибуты доступа менять нельзя. Переопределить метод в рамках одного класса нельзя, будем конфликт имен.

1. **Определите правило вызова переопределенных методов. Можно ли статические методы переопределить нестатическими и наоборот?**

**У переопределенного метода должны быть те же аргументы, что и у метода родителя.**

**У переопределенного метода должны быть тот же тип возвращаемого значения, что и у метода родителя.**

**Переопределять статический метод нельзя, вместо этого можно «скрыть» другой метод и написать новый с таким же наименованием.**

**8. Какие свойства имеют финальные методы и финальные классы? Как вы думаете, зачем их использовать?**

Финальные классы и финальные методы нельзя переопределять и наследоваться от них соответственно.

1. **Укажите правила приведения типов при наследовании. Напишите примеры явного и неявного преобразования ссылочных типов. Объясните, какие ошибки могут возникать при явном преобразовании ссылочных типов.**
2. **Что такое объект класса Class? Чем использование метода getClass() и последующего сравнения возвращенного значения с Type.class отличается от использования оператора instanceof?**

getClass возвращает значение класса объекта, а InstanceOf проверят находится ли он в одной цепочке наследования. То есть если есть class A и class B extends A, то выражения будут:

A.getClass.equls(B.getClass()) = false;

B instanceOf A = true;

**11. Укажите правила переопределения методов equals(), hashCode() и toString().**

**Рефлексивность** - для любого заданного значения x, выражение x.equals(x) должно возвращать true.  
*Заданного* — имеется в виду такого, что x != null

**Симметричность** - для любых заданных значений x и y, x.equals(y) должно возвращать true только в том случае, когда y.equals(x) возвращает true.

**Транзитивность** - для любых заданных значений x, y и z, если x.equals(y) возвращает true и y.equals(z) возвращает true, x.equals(z) должно вернуть значение true.

**Согласованность** - для любых заданных значений x и y повторный вызов x.equals(y) будет возвращать значение предыдущего вызова этого метода при условии, что поля, используемые для сравнения этих двух объектов, не изменялись между вызовами.

**Сравнение null** - для любого заданного значения x вызов x.equals(null) должен возвращать false

Переопределяем Equals: Boolean equals(Object obj){}

1. If (this == o) { return true; } Если ссылки равно, то и объекты сразу равны.
2. If(obj == null || this.getClass() != obj.getClass()) return false;

Далее приводим obj к нужному классу (на котором вызываем метод) И сравниваем поля поочередно.

**String** должен выводить всю информацию о объекте четко и наглядно.

**12. Что такое абстрактные классы и методы? Зачем они нужны? Бывают ли случаи, когда абстрактные методы содержат тело? Можно ли в абстрактных классах определять конструкторы? Могут ли абстрактные классы содержать неабстрактные методы? Можно ли от абстрактных классов создавать объекты и почему?**

13. Что такое интерфейсы? Как определить и реализовать интерфейс в java-программе? Укажите спецификаторы, которые приобретают методы и поля, определенные в интерфейсе. Можно ли описывать в интерфейсе конструкторы и создавать объекты? Можно ли создавать интерфейсные ссылки и если да, то на какие объекты они могут ссылаться?

**14. Для чего служит интерфейс Clonable? Как правильно переопределить метод clone() класса Object, для того, что объект мог создавать свои адекватные копии?**

15. Для чего служат интерфейсы Comparable и Comparator? В каких случаях предпочтительнее использовать первый, а когда – второй? Как их реализовать и использовать?