**1. Напишите (нарисуйте) алгоритм, описывающий порядок вызова конструкторов при создании объектов производного класса. (Постарайтесь предусмотреть все случаи)**

public class Car {

String model;

String country;

public Car(String model, String country) {

this. model = model;

this.country = country;

}

public class Honda extends Car {

String turboEngine;

public Honda(String model, String country, String turboEngine) {

super(model, country); //конструктор базового класса (сначала вызывается конструктор базового класса)

this. turboEngine = turboEngine; //потом вызывается конструктор производного класса

}

public static void main(String[] args) {

Honda honda = new Honda("Honda", "Japan", "Turbo Engine");

}

}

**2. Как вы думаете, зачем необходимо переопределение методов? Приведите примеры переопределенных методов. Может ли при переопределении метода меняться тип возвращаемого значения?**

Переопределение методов необходимо для того, чтобы в разных классах можно было вызывать один и тот же метод, но с разными действиями, теми действиями, которые у него уже есть в базовом классе, добавив ему новые действия, или изменив любое действие, которое уже есть в базовом.

class Car {

public void model(){

System.out.println(“any model”);

}

}

public class MyCar extends Car {

@Override

public void model() {

System.out.println("My car is Honda");

}

}

public class PreferrableCar {

public static void main(String[] args) {

MyCar myCar = new Car();

myCar.model();

}

}

Вернет значение “My car is Honda”, а не “any model”.

Тип возвращаемого значения возвращаться не может.

**3. Объясните, как вы понимаете смысл фразы “ссылка базового типа может ссылаться на объекты производных типов” и “объект подкласса может быть использован везде, где используется объект его суперкласса.”**

Ссылка базового класса может ссылаться на объекты как базового класса так и производного класса, в производном классе больше информации об объекте, т.к. он содержит в том числе и всю информацию из базового класса.

Если по ссылке базового типа вызвать метод, который переопределен в производном класса, он вызовется согласно типу объекта, а не ссылки.

**4. Попробуйте порассуждать на тему, наследуются или нет статические методы**

Статические методы наследуются, но не могут быть переопределены. Переопределяются только нестатические методы нестатическими.

**5. Как вы понимаете, то такое перегруженные методы? Приведите примеры перегруженных методов.**

Перегруженные методы, это методы, находящиеся в одном классе, имеющие одно имя и разные параметры

public Class A{

public m (int x){

}

public m (int y, int z){

}

public m(String test){

}

}

**6. Как вы думаете, зачем в языке Java надо применять ключевое слово final. Что может быть final в Java.**

Ключевое слово final запрещает наследование класса или методаю Final могут быть классы, методы и переменные

**7. Перечислите методы класса Object, которые вы знаете. Расскажите для чего они предназначены и как работают.**

класс Object родительский для всех классов

методы:

equals() – сравнивает между собой два объекта

hashCode() – возвращает хэш-код объекта

toString() – возвращает текстовое описание объекта

**8. Для чего следует применять метод equals(). Знаете ли вы “где” в Java метод equals() применяется неявно? Перечислите правила переопределения метода equals()**

Метод equals() применяется для сравнения двух объектов, по умолчанию сравнивает ссылки на объекты, поэтому для сравнения значений полей объектов метод нужно переопределять

1-проверяем не сравнивается ли объект сам с собой

2-проверяем не сравнивается ли объект с null

3-проверяем сравниваются ли объекты одного и того же типа

И затем можно приводить объект к типу класса, чтобы сравнивать значения по полям

**9. Что такое хэш-код? Что такое хэш-код объекта? Объясните, почему хэш-коды двух различных объектов могут совпасть. Перечислите известные вам правила переопределения метода hashCode()**

Метод hashCode() возвращает хэш-код объекта, должен переопределяться вместе с методом equals().

**10. Расскажите, когда применяется метод toString() и как его необходимо переопределять**

Метод toString() применяется для того, чтобы узнать содержимое того или иного объекта, возвращает строковое значение объекта. По умолчанию вернет хэш код объекта, поэтому для вывода необходимой информации необходимо переопределить метод toString(). Переопределить так, чтобы возвращал значения полей описания объекта, вызвавшего этот метод: getName(), getId() и другие