Вопросы по теме Методы и переменные:

1. Что такое local variable и instance variable?
2. Что такое сигнатура метода?
3. Что такое перегрузка метода?
4. Как передаются переменные в метод, по ссылке или по значению?
5. Какие бывают модификаторы доступа?
6. Как получить доступ к private члену класса?
7. Что такое рефлексия?
8. Может ли статический метод быть переопределен или перегружен?
9. Могут ли нестатические методы перегрузить статические?
10. О чем говорят ключевые слова «this», «super»?
11. Как получить доступ к переопределенному методу родительского класса?
12. Можно ли объявить метод абстрактным и статическим?
13. В чем разница между членом экземпляра класса и статическим членом класса?
14. Зачем нужны getter и setter, почему нельзя присваивать значения напрямую?

Переменные и методы.

Локальная переменная (local variable) – переменная, которая объявлена внутри метода. Существует в блоке кода с момента ее объявления, до конца блока кода.

Переменная экземпляра класса (instance variable) – переменная, которая объявлена внутри класса. Существует до того момента, пока существует объект класса.

Сигнатура метода – название метода и его параметры (точнее тип параметра) в определенном порядке.

Объявление метода – весь код, который описывается в методе.

Перегрузка метода – объявление нескольких методов в одном классе с одинаковыми именами, но с разными параметрами.

В Java переменные в метод **передаются по значению**. Для примитивных типов передается копия текущего объекта, а для ссылочных типов передается копия ссылки.

Ключевое слово **this** – нестатическая ссылочная переменная, которая ссылается на текущий экземпляр класса.

Ключевое слово **super** – нестатическая ссылочная переменная, которая ссылается на текущий экземпляр родительского класса.

Модификаторы доступа.

Модификатор доступа – ограничивает область видимости программных конструкций, к которым применяется. Может применяться к классам (кроме модификатора private), методам, конструктору класса, полям класса.

private – класс, внутри которого объявлен

package-private (default) – ↑ + пакет, в котором объявлен

protected – ↑ + классы-наследники

public – любое место программы

Доступа к private члену класса:

1. внутри класса доступ без ограничений;
2. вложенный класс имеет доступ;
3. с помощью getter и setter;
4. с помощью механизма рефликсии.

Рефлексия.

Рефлексия в Java – это механизм, который дает возможность:

- вносить изменения и получать информацию о классах, интерфейсах, полях и методах во время выполнения программы, при этом не зная имен этих классов, полей и методов;
- создавать новые экземпляры классов, вызывать у них методы, а также получать или устанавливать значения полей.

Работа с методами.

1. **Может ли статический метод быть переопределен или перегружен?**

   Статические методы можно перегружать.

   Статические методы нельзя переопределять.

2. **Могут ли нестатические методы перегрузить статические?**

Могут. Только получается два различных метода. Статический метод будет принадлежать классу (доступ через имя класса), а нестатический – конкретному объекту (доступ через вызов метода экземпляра класса).

3. **Как получить доступ к переопределенному методу родительского класса?**

   Через ключевое слово super.myMethod().

4. **Можно ли объявить метод абстрактным и статическим?**

Нельзя. Ключевое слово abstract указывает, что метод будет реализован в другом классе, а static принадлежит текущему классу и не может быть реализован в другом классе.

5. **В чем разница между членом экземпляра класса и статическим членом класса?**

Поля и методы экземпляра класса принадлежат конкретному объекту. Их вызов возможен только после предварительного создания объекта класса. Поля экземпляра класс могут иметь разные значения для каждого объекта.

Поля и методы статического класса принадлежат самому классу. Доступ с ним возможен без создания экземпляра класса (через имя класса). Статические поля инициализируются при инициализации класса. Статические методы имеют доступ только к статическим полям и методам и не могут использовать переменные this и super.

6. **Зачем нужны getter и setter, почему нельзя присваивать значения напрямую?**

В getter и setter мы можем сделать валидацию передаваемого / устанавливаемого значения. Например, без setter можно сделать так: person.age = -5, а с setter так:

```
public void setAge(int age) {
    if (age > 0 & age < 130) {  this.age = age;  }
}
```