МЕТОДЫ КЛАССА OBJECT

HASHCODE / EQUALS

CLONE

TOSTRING

В Java классы неявно наследуются от класса Object

Методы класса Object, доступные наследникам:

- int hashCode() возвращает хеш-код объекта. По умолчанию это рассчитанное специальным образом число на основании генератора случайных чисел
- boolean equals(Object obj) возвращает true, если объекты идентичны. По умолчанию выполняет сравнение по ссылке через оператор ==
- Object clone() создает и возвращает копию объекта. По умолчанию поверхностную копию

Методы класса Object, доступные наследникам (продолжение):

- String toString() возвращает строковое представление объекта. По умолчанию "имя класса" + "@" + "хеш-код объекта"
- Class getClass() - возвращает ссылку на класс объекта
- void wait(), void notify(), void notifyAll() используются в многопоточном программировании

Переопределение equals и hashCode

Метод equals используется для сравнения объектов, метод hashCode в основном используется для хранения объектов в коллекциях и мапах.

Для корректной работы данных методов их нужно переопределять в наследниках.

Правила переопределения equals

1. Рефлективность
   объект всегда равен самому себе: a.equals(a)
2. Симметричность
   если a.equals(b) , то и b.equals(a)
3. Транзитивность
   если a.equals(b), b.equals(c), то и a.equals(c)
4. Консистентность — сколько бы раз не вызывался equals без изменения состояния объекта, результат должен оставаться неизменным

Правила переопределения hashCode

1. Сколько раз бы не был вызван hashCode на объекте без изменения состояния, он должен возвращать то же значение. Значение может быть иным при следующем выполнении программы.
2. Если объекты равны по equals(), то hashCode() должен вернуть одинаковое значение для обоих объектов.
3. Если объекты не равны по equals(), hashCode() может возвращать одинаковые значения.

```java
public class Author {
    ...

    // методы сгененированы Idea

    // позволит сравнивать объекты методом equals
    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Author author = (Author) o;
        return age == author.age && Objects.equals(name, author.name)
    }
    // будет использоваться при добавлении объектов в hash таблицы
    @Override
    public int hashCode() {
        return Objects.hash(age, name);
    }
}
```

Правила переопределения clone

1. при необходимости нужно расширить модификатор доступа с protected на public
2. при необходимости нужно обработать исключение
3. при необходимости нужно изменить тип возвращаемого значения с Object на желаемый
4. реализация по умолчанию, копирует только значения примитивов, ссылочные свойства передаются в новый объект по ссылке
5. реализация по умолчанию (super.clone()) обязует класс реализовать интерфейс Cloneable

```java
public class Author implements Cloneable {
    ...
    @Override
    public Author clone() {
        try {
            // воспользовались реализацией по умолчанию
            return (Author) super.clone();
        } catch (CloneNotSupportedException e) {
            return null;
        }
    }
}
public class Book {
    ...
    @Override
    public Book clone() {
        // описали свою реализацию
        Book cloned = new Book();
        cloned.setTitle(title);
        cloned.setPageCount(pageCount);
        cloned.setAuthor(author.clone());
        return cloned;
    }
}
```