Java Professional
module #2

lecture #6 Comparing objects in Java. Comparator, Comparable. Method sort.

Mentor:

lecture #6. Comparing objects in Java. Comparator, Comparable. Method sort.

- Comparing objects
  - Операторы == и !=
  - Метод equals
- Comparator Interface
- Comparable Interface
- Comparable vs Comparator

Comparing objects

Примитивы

- Для примитивных типов - быть одинаковым означает иметь одинаковые значения!!!

Обьекты

- При сравнении с помощью == переменных-ссылок, указывающих на объекты, проверяется тот факт, что они указывают на один и тот же объект
- Стандартным методом, существующим у каждого класса, является метод equals

```
Cat one = new Cat();
Cat two = new Cat();

one.equals(two);
```

Метод equals(Obj obj) для класса Cat

Comparator Interface

- Интерфейс компаратора используется для упорядочения объектов определяемых пользователем.

- Объект сравнения способен сравнивать два объекта одного и того же класса

```
public int compare(Object obj1, Object obj2)
```

Задача: у нас есть ArrayList котов, содержащий такие поля: имя, год рождения, вес и тд. нам нужно отсортировать массив на основе года рождения, а если год одинаковый то по имени.

Способ 1. - написать собственную функцию sort(), используя известный алгоритм сортировки.
Способ 2. - использование Comparator Interface

Comparator Interface

Как работает метод sort() класса Collections?

- Внутри метод Sort вызывает метод Compare сортируемых классов.
- Чтобы сравнить два элемента, он спрашивает: «Какой из них больше?»
- Метод сравнения возвращает -1, 0 **или** 1, чтобы сказать, **меньше**, **равно или больше** другого.
-
- Он использует этот результат, чтобы затем определить, следует ли их поменять местами для их сортировки.

Comparable Interface

- Интерфейс Comparable используется для сравнения объекта того же класса с экземпляром этого класса, он обеспечивает упорядочение данных для объектов пользовательского класса.
- Класс должен реализовать интерфейс java.lang.Comparable для сравнения своего экземпляра, он предоставляет метод compareTo, который принимает параметр объекта этого класса.

Задача 1

Дан массив пар, состоящий из двух полей строкового и целочисленного типов, отсортировать массив в возрастающем лексикографическом порядке, и если две строки одинаковы, отсортируйте их на основе их целочисленного значения.

Input: {{"abc", 3}, {"a", 4}, {"bc", 5}, {"a", 2}}
Output: {{"a", 2}, {"a", 4}, {"abc", 3}, {"bc", 5}}

Comparable Interface

Задача 2 – самостоятельно

Дан массив пар, состоящий из двух строк с именами и фамилиями, отсортировать массив в возрастающем лексикографическом порядке имени, и если две строки одинаковы, отсортируйте их по фамилии.

Input: {{"abc", "last"}, {"pklz", "yelp"}, {"rpng", "note"}, {"ppza", "xyz"}}
Output: {{"abc", "last"}, {"pklz", "yelp"}, {"ppza", "xyz"}, {"rpng", "note"}}

Comparable vs Comparator

Java предоставляет два интерфейса для сортировки объектов с использованием элементов данных класса:

1. Comparable (сопоставимый)
2. Comparator

Сопоставимый объект способен сравнивать себя с другим объектом.
Сам класс должен реализовывать интерфейс java.lang.Comparable для сравнения своих экземпляров.
Интерфейс Comparable содержит один единственный метод int compareTo(E obj), который сравнивает текущий объект с объектом, переданным в качестве параметра.
Если в классе не реализован Comparable или реализация не подходит
есть Интерфейс Comparator, предполагающий применение сравнения по разным характеристикам.
Подводя итог:
если сортировка объектов должна быть основана на естественном порядке, используйте Comparable,
если вам необходимо выполнить сортировку по характеристикам объектов, используйте Comparator.