

Некоторые стандартные интерфейсы Java и примеры их использования

Семинар 3








Содержание урока



Что будет на уроке сегодня

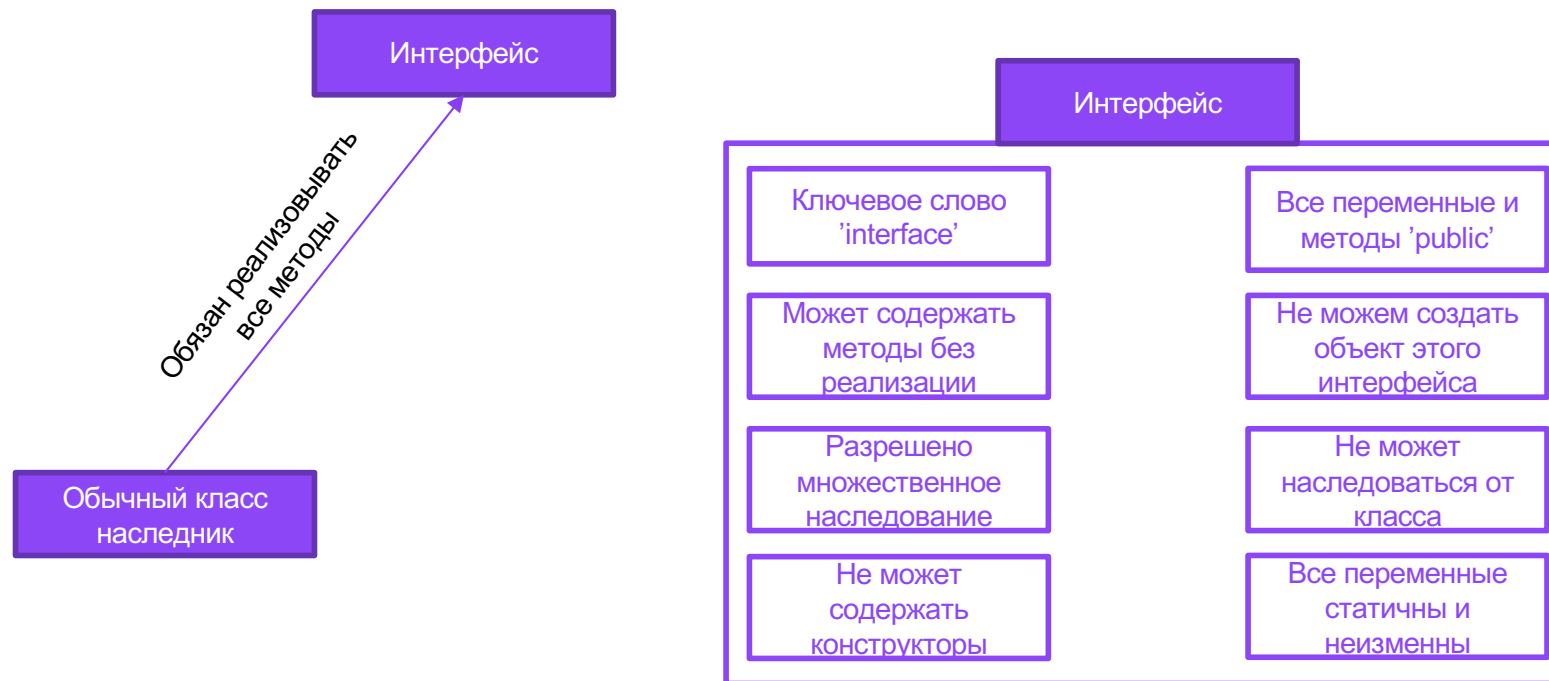
-  Повторим теорию, пройденную на предыдущем семинаре - интерфейсы
-  Получим практические навыки в использовании некоторых стандартных интерфейсов Java
-  Изучим примеры использования стандартных интерфейсов в различных ситуациях



Повторим теоретические моменты и
попрактикуемся



Что такое интерфейс?



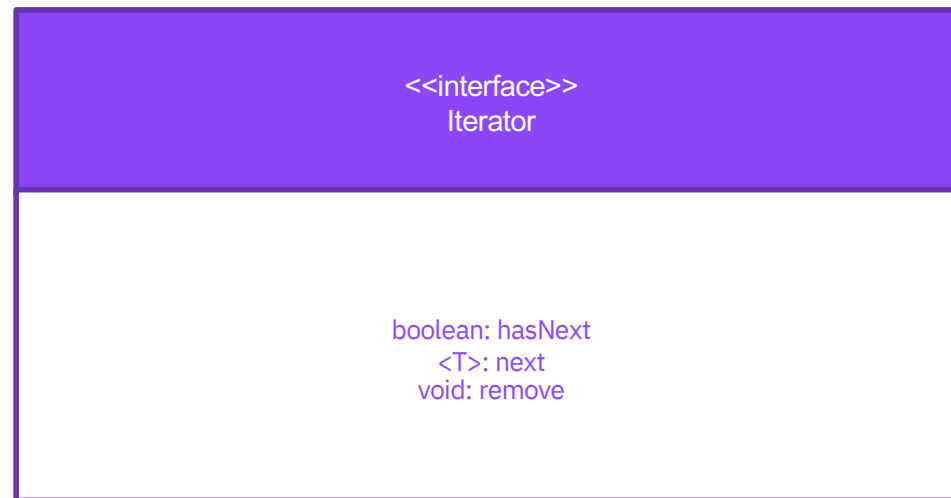


Интерфейсы маркеры









Iterator



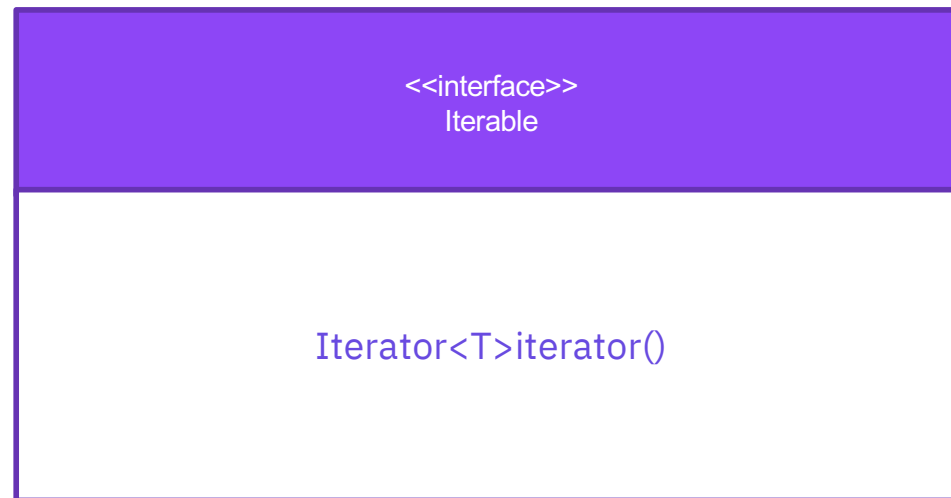


Задача 1 - Iterator

-  Создать класс Студент
-  Создать класс УчебнаяГруппа
-  Создать класс УчебнаяГруппаИтератор, заставив его реализовать интерфейс Iterator
-  Реализовать его контракты (включая удаление)







Iterable





Задача 2 - Iterable

-  Модифицировать класс УчебнаяГруппа, заставив его реализовать интерфейс Iterable
-  Реализовать метод iterator() возвращающий экземпляр созданного нами итератора
-  Создать класс УчебнаяГруппаСервис, добавив в него метод удаления студента по ФИО
-  Создать класс Контроллер, добавив в него метод удаления студента и вызывать в нем созданный метод из УчебнаяГруппаСервис



Comparable

<<interface>>
Comparable

int compareTo(O o)

```
@Override  
public int compareTo(Int x) {  
    if ( x > this.x ) {  
        return -1;  
    } else if ( x < this.x ) {  
        return 1;  
    } else return 0;  
}
```

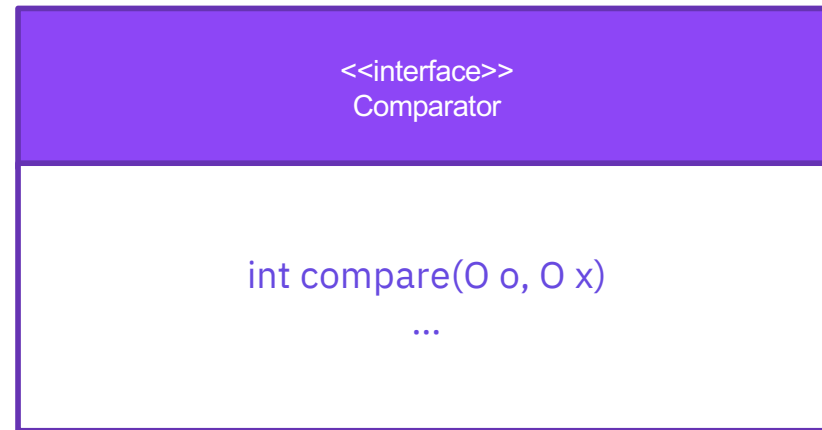


Задача 3 - Comparable

- 📌 Модифицировать класс Студент, заставив его реализовать интерфейс Comparable
- 📌 Реализовать контракт compareTo () со сравнением по какому-либо параметру (пример studentId)
- 📌 Модифицировать класс УчебнаяГруппаСервис, добавив в него метод сортировки списка студентов по id
- 📌 Модифицировать класс Контроллер, добавив в него метод сортировки списка студентов по id и вызывать в нем созданный метод из УчебнаяГруппаСервис



Comparator



```
class PersonAgeComparator implements Comparator<Person>{
    public int compare(Person a, Person b){
        if(a.getAge() > b.getAge())
            return 1;
        else if(a.getAge() < b.getAge())
            return -1;
        else
            return 0;
    }
}
```



Задача 4 - Comparator






- 📌 Создать класс СтудентКомпаратор реализующий интерфейс Comparator
- 📌 Реализовать контракт compare () со сравнением по какому-либо параметру (пример: сочетание Имя+Фамилия)
- 📌 Модифицировать класс УчебнаяГруппаСервис, добавив в него метод сортировки списка студентов по ФИО
- 📌 Модифицировать класс Контроллер, добавив в него метод сортировки списка студентов по ФИО и вызывать в нем созданный метод из УчебнаяГруппаСервис



Домашнее задание



Домашнее задание

-  Создать класс Поток содержащий в себе список УчебныхГрупп и реализующий интерфейс Iterator
-  Создать класс StreamComparator, реализующий сравнение количества групп входящих в Поток
-  Создать класс ПотокСервис, добавив в него метод сортировки списка потоков, используя созданный StreamComparator
-  Модифицировать класс Контроллер, добавив в него созданный сервис
-  Модифицировать класс Контроллер, добавив в него метод сортирующий список потоков, путем вызова созданного сервиса



Подведем итоги



Спасибо за работу!