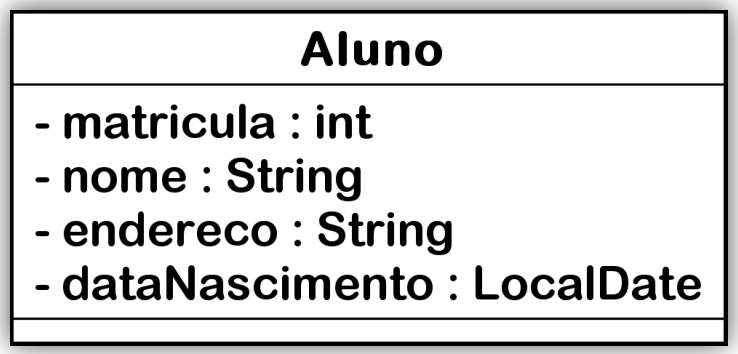
Ordenação de coleções

FURB
UNIVERSIDADE DE BLUMENAU

Ordenação de objetos de uma coleção

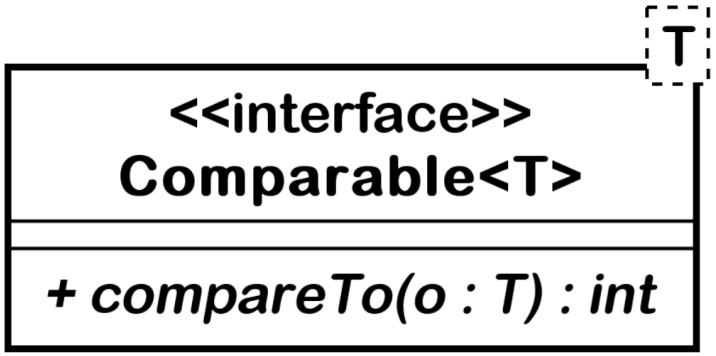
- Numa lista contendo objetos da classe **`Aluno`**, quais atributos devem ser usados como critério para ordenação?



- Existem dois conceitos de ordenação em Java:
  - Ordenação **natural** – é a ordenação típica de objetos de uma determinada classe. Cada classe possui apenas uma ordenação natural;
  - Ordenação **artificial** – uma forma alternative de ordenar objetos de uma classe. Cada classe pode possuir várias ordenações artificais

Definindo a ordenação natural

- A ordenação natural de objetos de uma classe é especificada ao implementar a interface **`Comparable`**:



- O método **`compareTo()`** deve retornar:
  - **`-1`**: se **this < outro**
  - **`0`**: se **this** for igual à **outro**
  - **`+1`**: se **this > outro**

Exemplo

```
public class Aluno implements Comparable<Aluno> {

    private int matricula;
    private String nome;
    private String endereco;
    private LocalDate dataNascimento;


    @Override
    public int compareTo(Aluno o) {
        if (this.matricula < o.matricula)
            return -1;
        if (this.matricula > o.matricula)
            return +1;
        return 0;
    }

}
```

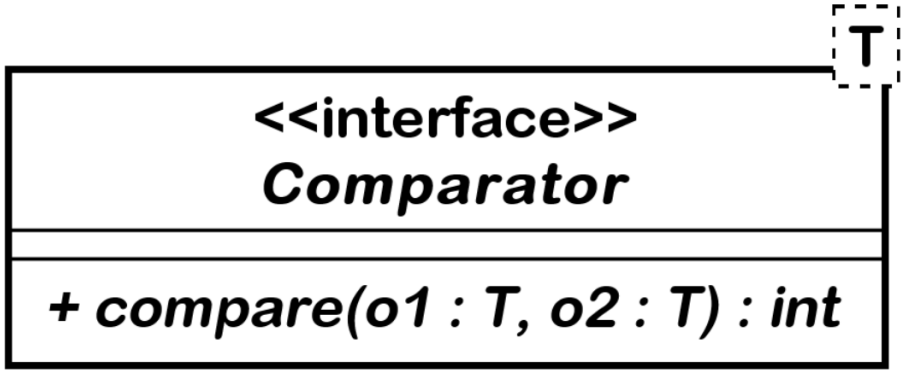
Ordenando uma coleção

- Para ordenar coleções, utilizar o método **`sort()`** da classe **`Collections`**.

```
ArrayList<Aluno> alunos = new ArrayList<>();

alunos.add(new Aluno(1010,"José","R.XV de Novembro 192",
LocalDate.of(2000,2,18)));

alunos.add(new Aluno(6925,"Marta","R. 7 de setembro,
942",LocalDate.of(2001,9,7)));

Collections.sort(alunos);
```

Definindo a ordenação artificial

- A ordenação artificial de objetos de uma classe é especificada ao implementar a interface **`Comparator`**:



- O método **`compare()`** deve retornar:
  - **`-1`**: se **o1 < o2**
  - **`0`**: se **o1** for igual à **o2**
  - **`+1`**: se **o1 > o2**

Exemplo

```java
import java.util.Comparator;

public class OrdenacaoPorNome implements Comparator<Aluno> {

    @Override
    public int compare(Aluno o1, Aluno o2) {
        return o1.getNome().compareTo(o2.getNome());
    }

}
```

```java
ArrayList<Aluno> alunos = new ArrayList<>();
alunos.add(new Aluno(1010,"Zé","R.XV de Novembro 192",LocalDate.of(2000,2,18)));
alunos.add(new Aluno(6925,"Marta","R. 7 de setembro, 942",LocalDate.of(2001,9,7)));

Collections.sort(alunos, new OrdenacaoPorNome());
```

Exemplo

```
import java.util.Comparator;

public class OrdenacaoPorNomeMatricula implements Comparator<Aluno> {

    @Override
    public int compare(Aluno o1, Aluno o2) {
        int ordem = o1.getNome().compareTo(o2.getNome());
        if (ordem == 0) {
            if (o1.getMatricula() < o2.getMatricula()) {
                ordem = -1;
            } else if (o1.getMatricula() > o2.getMatricula()) {
                ordem = +1;
            } else {
                ordem = 0;
            }
        }

        return ordem;
    }

}
```