DIGITAL INNOVATION ONE

Comparators

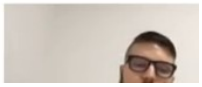
- ✓ Interfaces que aprenderemos
  - ✓ java.util.Comparator - Interface para definir classe com regra de ordenação
  - ✓ Java.util.Comparable - Interface para definir regra de ordenação em uma classe de domínio
- ✓ Algoritmos de ordenação
- ✓ Utilizado primariamente em java.util.List
- ✓ Permite a ordenação de objetos complexos (criados pelo usuário)

DIGITAL INNOVATION ONE

Exercício final

1 – Crie uma Lista de um objeto complexo e execute as seguintes operações:

- Adicione elementos nesta lista.
- Ordene implementando a interface java.util.Comparator no seu objeto complexo.
- Ordene implementando um novo objeto com a interface java.util.Comparable
- Ordene usando uma expressão lamda na chamada de suaLista.sort()
- Ordene usando referências de métodos e os métodos estáticos de Comparator
- Ordene coleções TreeSet e TreeMap



Collections V - Comparators

Dado o ArrayList de estudantes: [{nome="Matheus", idade=22}, {nome="Carla", idade=65}, {nome="Joice", idade=15}, {nome="Talison", idade=19}], responda:

Após implementar Comparable na classe Estudante, qual o estado resultante após a ordenação com base na regra 'this.idade - other.idade'?

- [{nome="Joice", idade=15}, {nome="Talison", idade=19}, {nome="Matheus", idade=22}, {nome="Carla", idade=65}] ✔
- [{nome="Matheus", idade=22}, {nome="Carla", idade=65}, {nome="Joice", idade=15}, {nome="Talison", idade=19}]
- Erro de execução ou compilação. ✖
- [ ]
- [{nome="Carla", idade=65}, {nome="Matheus", idade=22}, {nome="Talison", idade=19}, {nome="Joice", idade=15}]

PRÓXIMA PERGUNTA

Collections V - Comparators

Dado o ArrayList de estudantes: [{nome="Matheus", idade=22}, {nome="Carla", idade=65}, {nome="Joice", idade=15}, {nome="Talison", idade=19}], responda:

Referente ao método "Comparator.reversed", assinale a alternativa correta:

- Não recebe parâmetros, retorna um inteiro para a ordem do objeto atual na ordem reversa com base no Comparator da instância atual (this).
- Este método não existe. ✖
- Não recebe parâmetros, retorna a lista em ordem reversa com base no Comparator da instância atual (this).
- Não recebe parâmetros, retorna um comparator na ordem reversa com base no Comparator da instância atual (this). ✔
- Recebe como argumento um objeto do tipo Comparator que compara objetos na ordem inversa ao objeto atual (this).

PRÓXIMA PERGUNTA

Collections V - Comparators

Dado o ArrayList de estudantes: [{nome="Matheus", idade=22}, {nome="Carla", idade=65}, {nome="Joice", idade=15}, {nome="Talison", idade=19}], responda:

Referente ao método "Comparable.compareTo", assinale a alternativa correta:

- Recebe um argumento genérico, e retorna um número inteiro. Se positivo, o objeto atual (this) será antes que o objeto de comparação. Se negativo, o objeto atual (this) será depois do objeto de comparação. Se zero, os objetos permanecem na ordem de inserção. ✔
- Recebe um argumento genérico, e retorna um número inteiro. Se positivo, o objeto atual (this) será depois que o objeto de comparação. Se negativo, o objeto atual (this) será antes do objeto de comparação. Se zero, os objetos na ordem de inserção.
- Recebe um argumento genérico, e retorna um número inteiro. Se positivo, o objeto atual (this) será depois que o objeto de comparação. Se negativo, o objeto atual (this) será antes do objeto de comparação. Se zero, os objetos invertem a ordem de inserção.
- Este método não existe.
- Recebe um argumento genérico, e retorna um número decimal. Se positivo, o objeto atual (this) será antes que o objeto de comparação. Se negativo, o objeto atual (this) será depois do objeto de comparação. Se zero, os objetos invertem a ordem de inserção.

PRÓXIMA PERGUNTA

Collections V - Comparators

Dado o ArrayList de estudantes: [{nome="Matheus", idade=22}, {nome="Carla", idade=65}, {nome="Joice", idade=15}, {nome="Talison", idade=19}], responda:

Referente ao método "Comparator.compare", assinale a alternativa incorreta::

- Recebe dois argumentos genéricos e retorna o maior objeto. ✖
- Recebe dois argumentos genéricos e retorna o menor objeto.
- Recebe dois argumentos genéricos e retorna um inteiro. ✔
- Recebe um argumento genéricos e retorna um inteiro.
- Este método não existe.

PRÓXIMA PERGUNTA

Collections V - Comparators

Dado o ArrayList de estudantes: [{nome="Matheus", idade=22}, {nome="Carla", idade=65}, {nome="Joice", idade=15}, {nome="Talison", idade=19}], responda:

Após implementar Comparable na classe Estudante, qual o estado resultante após a ordenação com base na regra 'other.idade - this.idade'?

- Erro de execução ou compilação. ✖
- [{nome="Carla", idade=65}, {nome="Matheus", idade=22}, {nome="Talison", idade=19}, {nome="Joice", idade=15}] ✔
- [ ]
- [{nome="Joice", idade=15}, {nome="Talison", idade=19}, {nome="Matheus", idade=22}, {nome="Carla", idade=65}]
- [{nome="Matheus", idade=22}, {nome="Carla", idade=65}, {nome="Joice", idade=15}, {nome="Talison", idade=19}]

PRÓXIMA PERGUNTA

Collections V - Comparators

Assinale a alternativa com código que compila:

- Collections.sort(Set.of(1, 5, 0, 9, 12, 23), new MapComparator());
- Collections.sort(Set.of(1, 5, 0, 9, 12, 23), new SetComparator());
- Collections.sort(Map.of("one", 1, "five", 5, "zero", 0, "nine", 9, "twelve", 12, "twenty-three", 23));
- Collections.sort(Set.of(1, 5, 0, 9, 12, 23));
- Collections.sort(List.of(1, 5, 0, 9, 12, 23)); ✔

FINALIZAR

Collections V - Comparators

Dado o ArrayList de estudantes: [{nome="Matheus", idade=22}, {nome="Carla", idade=65}, {nome="Joice", idade=15}, {nome="Talison", idade=19}], responda:

Assinale a alternativa correta:

- java.util.Comparable não deve ser usada na implementação da classe de domínio pois recebe dois argumentos como parâmetro do método 'compareTo'
- java.util.Comparable deve ser usada na implementação da classe de domínio pois recebe dois argumentos como parâmetro do método 'compareTo' ✖
- java.util.Comparable deve ser usada na implementação da classe de domínio pois recebe apenas um argumento como parâmetro do método 'compareTo' ✔
- java.util.Comparator deve ser usada na implementação da classe de domínio pois recebe dois argumentos como parâmetro do método 'compareTo'
- java.util.Comparator deve ser usada na implementação de uma classe de negócio pois recebe apenas um argumento como parâmetro do método 'compareTo'

PRÓXIMA PERGUNTA

Collections V - Comparators

Dado o ArrayList de estudantes: [{nome="Matheus", idade=22}, {nome="Carla", idade=65}, {nome="Joice", idade=15}, {nome="Talison", idade=19}], responda:

Assinale a alternativa correta:

- O método Collections.sort() sempre requer um argumento com uma lista de tipo genérico que estenda Comparable. ✖
- O método Collections.sort() é sobrecarregado, pode receber tanto uma lista de tipo genérico que estenda Comparable, como também uma lista comum sendo que o segundo parâmetro é uma implementação de Comparable. ✔
- Nenhuma das alternativas está correta.
- O método Collections.sort(estudantes) não requer um argumento com uma lista de tipo genérico que estenda Comparable.

PRÓXIMA PERGUNTA

Collections V - Comparators

Dado o ArrayList de estudantes: [{nome="Matheus", idade=22}, {nome="Carla", idade=65}, {nome="Joice", idade=15}, {nome="Talison", idade=19}], responda:

Referente ao método "Collection.sort()" assinale a alternativa correta:

- Recebe uma instância de List com tipo genérico em Comparable, e retorna na ordem definida pela implementação de compareTo.
- Recebe uma instância de Collection e retorna na ordem natural do objeto.
- Recebe uma instância de Collection com tipo genérico em Comparator, e retorna na ordem definida pela implementação de compareTo.
- Este método não existe. ✔
- Recebe uma instância de Collection com tipo genérico em Comparable, e retorna na ordem definida pela implementação de compareTo.

PRÓ

Collections V - Comparators

Dado o ArrayList de estudantes: [{nome="Matheus", idade=22}, {nome="Carla", idade=65}, {nome="Joice", idade=15}, {nome="Talison", idade=19}], responda:

Não é uma chamada válida para o método 'estudantes.sort':

- estudantes.sort(Comparator.comparingInt(Estudante::getIdade).reversed())
- estudantes.sort((first, second) -> first.getIdade() - second.getIdade())
- estudantes.sort((first, second) -> second.getIdade() - first.getIdade())
- estudantes.sort((first, second) -> first.getIdade() - second.getIdade()).reversed() ✔
- estudantes.sort(Comparator.comparingInt(Estudante::getIdade))

PRÓXIMA PERGUNTA