PROJETO ENTRA21

Lista de Exercícios2 - Java Avançado
COLLECTIONS

1) Abra sua classe Conta e veja se ela possui o atributo numero. Se não possuir, adicione-o e crie o seu getter.

2) Faça sua classe ContaPoupanca implementar a interface Comparable<ContaPoupanca>. Utilize o critério de ordenar pelo número da conta ou pelo seu saldo.

3) Crie uma classe TestarOrdenacao, instancie várias contas e adicione-as a uma List<ContaPoupanca>. Use o Collections.sort() nesta lista. Faça um laço para imprimir todos os números das contas na lista já ordenada:

ATENÇÃO: observe que temos um método compareTo em nossa classe e nosso código nunca o invoca! Isto é muito comum. Reescrevemos (ou implementamos) um método e quem o invocará será um outro conjunto de classes (nesse caso, quem está chamando o compareTo é o Collections.sort, que o usa como base para o algoritmo de ordenação). Isso cria um sistema extremamente coeso e, ao mesmo tempo, com baixo acoplamento: a classe Collections nunca imaginou que ordenaria objetos do tipo ContaPoupanca, mas já que eles são Comparable, o seu método sort está satisfeito.

4) O que teria acontecido se a classe ContaPoupanca não implementasse Comparable<ContaPoupanca> mas tivesse o método compareTo? Faça um teste e veja o que acontece. Basta ter o método, sem assinar a interface?

5) Utilize uma LinkedList em vez de ArrayList. Precisamos alterar mais algum código para que essa substituição funcione? Rode o programa. Alguma diferença?

6) Como posso inverter a ordem de uma lista?

7) Mude o critério de comparação da sua ContaPoupanca. Adicione um atributo nomeDoCliente na sua classe (caso ainda não exista algo semelhante), mude o compareTo para que uma lista de ContaPoupanca seja ordenada alfabeticamente pelo atributo nomeDoCliente.

8) Insira novas contas através de um laço atribuindo saldos aleatórios com um objeto (java.util.Random) altere a ordenação para o saldo. Teste a ordenação. Dica: utilize um array ContaPoupanca[ ] e utilize em random nextDouble * 2000