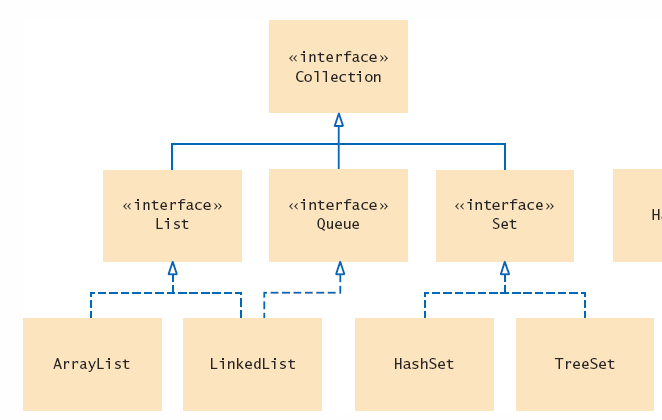
**Arquitetura de Software**

**Prof. Bernardo Copstein**

# Exercício sobre padrões de criação

1. Considere o seguinte problema: uma aplicação faz uso frequente de coleções. Tipicamente itera-se sobre os elementos armazenados visitando-os. Dependendo do momento algumas características de uso podem variar:
   * O tamanho da coleção é conhecido de antemão e deseja-se visitar os elementos na ordem que foram inseridos
   * O tamanho da coleção é desconhecido de antemão e deseja-se visitar os elementos na ordem em que foram inseridos
   * O tamanho da coleção é desconhecido de antemão, a ordem de iteração não é importante e não se admitem elementos repetidos na coleção
   * O tamanho da coleção é desconhecido, não se admitem elementos repetidos na coleção e a iteração deve seguir uma ordenação natural dos elementos.

Explorando os padrões “Factory” e “Singleton” e a API de coleções de Java, defina uma solução para este problema. Explique os padrões de projeto utilizados e apresente um exemplo de uso.



1. Em um determinado banco cada cliente possui dois saldos: o saldo livre para movimentações de depósitos e retiradas e o saldo em investimentos. Para poder movimentar o saldo investido o cliente deve solicitar que todo ou parte do valor investido seja transferido para o saldo livre com 2 dias de antecedência. Cada cliente – de acordo com seu perfil – tem uma taxa de remuneração para os investimentos. Da mesma forma tem uma taxa de juros que é cobrada caso seu saldo livre fique negativo (o banco admite saques a descoberto até um certo limite). Um desenvolvedor criou uma classe Java para modelar esta conta bancária, porém, o uso do seu método construtor tornou-se um tormento. São muitos os parâmetros e na verdade apenas o número da conta e o nome do correntista são necessários na abertura. Os demais podem ou não ser indicados na abertura ou mais tarde visto que podem ser utilizados valores padrão (os saldos iniciais podem ser zero assim como as taxas e o limite para saques a descoberto; a categoria inicial pode ser Categoria.Normal). Utilize o padrão Builder para melhorar a usabilidade na criação de instâncias desta classe. Refatore ela usando este padrão e apresente diferentes exemplos de uso.

|  |
| --- |
| enum Categoria { NORMAL, ADVANCED, PREMIUM };  public class Conta{  ...  public Conta(Long numero, String nomeCorrentista,  int saldoLivreInicial, int salAplicacaoInicial,  double taxaRemuneracao, double taxaSaldoNegativo,  Categoria categoriaInicial){  ...  }  } |

1. Em um jogo, a classe “Personagem” tem 3 atributos que devem ser informados na hora da criação da instância: visibilidade [0,10], poder [0;100] e vidas [0,5]. Aproximadamente 60% dos personagens criados são os ditos “normais” que são criados com visibilidade = 10, poder = 50 e vidas = 3. Uns 15% são chamados de “poderosos”. Nestes a visibilidade é 10, as vidas são 3, mas o poder é ajustado na hora da criação. Outros 15% são os conhecidos como “soturnos” onde o poder é 50, as vidas são 3 e a visibilidade é ajustada na hora da criação. Finalmente os últimos 10% tem os três parâmetros configurados na hora da criação. Explore o padrão “static factory method” na construção desta classe. Escreva a classe e exemplos de uso.
2. Explique se você considera que os padrões propostos para os exercícios 1, 2 e 3 eram realmente os mais adequados para cada uma das situações. Justifique sua resposta.