	**UNIVERSIDADE FEDERAL DA PARAIBA**	
DEPARTAMENTO	**SISTEMAS E COMPUTAÇÃO**	
DISCIPLINA	**MÉTODOS E PROJETO DE SOFTWARE**	
PROFESSOR	**Raoni Kulesza**	

Exercícios sobre Coleções

Questão 1 - Crie um projeto no Eclipse de nome ProjBanco. Crie um pacote banco para armazenar as classes. Crie duas classes Conta e Cliente. A estrutura de cada uma é descrita a seguir:

a. A classe Conta deve ter dois atributos numero (String) e saldo (double) e os métodos para setar e retornar o número (setNumero() e getNumero()), pegar o saldo (getSaldo()), debitar (debitar(valor)) e creditar (creditar(valor)) valores na conta. Implemente ainda dois construtores, sendo um construtor default que inicializa o numero igual "7777" e saldo com valor igual a 77 Crie outro construtor que recebe o número da conta e o valor do saldo inicial como parâmetro.
b. A classe Cliente deve ter dois atributos cpf e nome. Implemente os respectivos métodos get/set. Implemente um construtor default que inicializa o cliente com cpf igual "111.111.111-77" e nome "Nome e Sobrenome" e outro construtor que receba o cpf e o nome como parâmetros.
c. Implemente o método toString() para cada uma das classes. Este método deverá retornar uma String contendo os atributos da classe.

Questão 2 - Crie uma classe Banco dentro do pacote banco. Esta classe deverá manter um cadastro de contas e clientes associados. Um cliente pode ter mais de uma conta. Mas uma conta é única no banco e pertence apenas a um único cliente. Para isso utilize um atributo cadastro (Interface Map) para armazenar as contas e seus respectivos clientes (Note que a chave do mapa será a Conta!).

a. Implemente um construtor default Banco() que inicializa o mapa.
b. Implemente os seguintes métodos na classe Banco
   i. inserir(Conta c, Cliente c): recebe uma conta e um cliente e insere-os no cadastro. Esse método não retorna nada (void)
   ii. buscaConta(String numero): retorna o objeto conta correspondente ao numero passado como parâmetro. Caso não exista uma conta com esse número, o método deve retornar nulo.
   iii. buscaCliente(String cpf): retorna o objeto cliente correspondente ao cpf passado como parâmetro. Caso não exista um cliente com esse cpf, o método deve retornar nulo.
   iv. buscaContasDeUmCliente(String cpf): retorna uma lista das contas pertencentes a um mesmo cliente. Caso o cliente não exista, retorna uma lista vazia.
c. Crie uma classe TestaBanco com um método main para testes. Crie 6 contas e 4 clientes diferentes. Insira-os no banco. Note que alguns clientes ficarão com mais de uma conta. Teste os métodos de busca implementados. Imprima os resultados.

*OBS: Para implementar os métodos de busca, você terá que fazer uso da interface **Iterator** para percorrer os elementos da coleção ou o comando for each .*

Questão 3 - Crie um projeto no Eclipse com o nome **ProjAgenda**. Crie um pacote **agenda**. Nesse pacote crie uma classe **Pessoa**. Esta classe deve ter: (1) propriedades: **nome**, **telefone e endereco** encapsuladas e com seus respectivos métodos **get/set;** (2) um **construtor default** que inicializa uma **Pessoa** com nome *Nome e Sobrenome,* telefone *7777-1111* e endereço *Logradouro Xpto*, respectivamente. Crie também um **outro construtor** que inicialize a pessoa com base em parâmetros fornecidos, (3) método **toString** implementado que imprima o nome da pessoa, seguido de seus dados de telefone e endereço. Em seguida faça as seguintes modificações:

a. Crie uma classe **Agenda**. Esta classe terá um atributo privado ***contatos*** que será um mapa (*Interface Map*) para armazenar os contatos (Pessoas), onde a chave para encontrar uma pessoa nesse mapa será o ***nome***. Em seguida, implemente um construtor **default** da classe **Agenda** que inicializa o mapa ***contatos***.
b. Implemente na classe **Agenda** o método ***buscaPessoa*** que recebe como parâmetro o ***nome*** e retorna a pessoa encontrada. c. Implemente na classe **Agenda** o método ***inserePessoa*** para adicionar uma pessoa na lista de contatos. Esse método recebe como parâmetro o objeto **Pessoa** a ser adicionado. *OBS: lembre-se que o nome da pessoa é a chave do objeto no mapa* ***contatos***
d. Implemente na classe **Agenda** o método ***listarNomes*** que retorna uma lista com APENAS o nome das pessoas cadastradas (*lista das chaves*). Em seguida, implemente outro método ***listarPessoas*** que retornará uma lista dos objetos Pessoa cadastrados (*lista dos valores*).
e. Crie uma classe **UsaAgenda** para testes dos métodos implementados. Crie 6 objetos Pessoa com dados diferentes e os insira na agenda. Depois, imprima todos os nomes da agenda. Em seguida, imprima todos os elementos da agenda (dados completos das pessoas). Por fim, faça uma busca por um nome específico. *OBS: Note que o atributo* ***contatos*** *da classe* ***Agenda*** *é privado, portanto você não pode acessá-lo diretamente!*

Questão 4 - Crie um projeto no Eclipse com o nome **ProjMapa**. Crie um pacote mapas. Nesse pacote crie uma classe **UsaMapas** que conterá apenas um método main para testar o uso das classes que implementam a interface **Map**. A seguir resolva os seguintes itens:

a. Crie um mapa de **Strings** (**Map<String, String>**) que terá como chave o login (**String**) de um usuário, e o valor será a senha (**String**) do usuário. OBS: Use a classe **HashMap** para instanciar esse mapa.
b. Insira nesse mapa, três usuários com os seguintes logins: ronaldo, romario e roberto. Insira senhas quaisquer para cada usuário.
c. Imprima os elementos desse mapa. Verifique a ordem que foram impressos.
d. Crie outro mapa, agora com a classe **TreeMap**, onde os usuários sejam impressos ordenados pelo login. Utilize a abordagem **java.lang.Comparable** Insira os mesmos usuários da questão b. Imprima esse novo mapa e verifique a ordem em que foram impressos. OBS: Vejas os slides para saber qual classe implementa um mapa.ordenado.

Questão 5 – Crie um projeto no Eclipse com o nome **ProjPessoa**. Nesse projeto realize os seguintes passos:

a. Crie uma classe **Data** com os seguintes atributos dia, mês e ano todos do tipo **int**. A data deve ter o formato DD /MM /AA
b. Crie uma classe **Pessoa** com os seguintes atributos: cpf do tipo **String**, nome do tipo **String** e data_nascimento do tipo **Data**, criada no item anterior;
c. Crie uma classe **TestaComparator** que deve conter duas **TreeSet**. A primeira ordena as pessoas pelo nome e se duas pessoas têm nomes iguais, ordena pelo CPF. A segunda ordena pela data de nascimento. Utilize a abordagem da interface **java.lang.Comparator** e crie duas classes **ComparadorNome** e **ComparadorData** que implementam essa interface e contenham os algoritmos de ordenação propostos.
d. Adicione código na classe **TestaComparator** para testar as implementações das ordenações