**UNIVERSIDADE ESTADUAL DA PARAÍBA**

**CENTRO DE CIÊNCIA E TECNOLOGIAS**

**CURSO DE CIÊNCIA DA COMPUTAÇÃO**

**DISCIPLINA:** LINGUAGEM DE PROGRAMAÇÃO II

**3ª VERIFICAÇÃO DE APRENDIZAGEM**

**Observações – leia antes de começar a prova:**

* **Fiquem atentos a sintaxe da linguagem e a utilização correta do paradigma orientado a objetos, à medida que o semestre avança, espera-se uma maior evolução e maturidade por parte do aluno nos conceitos e linguagem, portanto o nível de exigência e cobrança aumenta por parte do professor.**
* **Nas questões onde é necessário implementar arrays (vetores ou matrizes), não será considerado o código de quem implementar usando Collections, por exemplo ArrayList, ou a classe Array do Java, sendo pontuada com zero (0,0) de imediato a questão. O objetivo da questão não é o uso de classes/interfaces prontas.**

1. Implemente uma classe Revista com os atributos String nome, double valor e String autor. Essa classe deve ter os construtores Revista(String nome) e Revista(String nome, double valor, String autor). Implemente um método que imprime na tela as informações de Revista.

Além da classe Revista, implemente uma classe Colecao, que possui um nome e um conjunto de revistas. A classe deve ter um construtor que inicializa um conjunto de revistas. Os métodos que deve ser implementados são

* 1. void incluirRevista(Revista revista): inclui uma revista da coleção de revistas. Observe que nenhum tamanho de array foi indicado, ou seja, o seu array deve crescer dinamicamente.
  2. void excluirRevista(Revista revista): remove uma revista da coleção de revistas
  3. void exibirColecao(): imprime na tela uma coleção de revistas

Implemente uma classe de aplicação que crie uma coleção de revistas, inclua algumas revistas, exiba todas as revistas, exclua uma revista das revistas incluídas e exiba a coleção após a exclusão. **(4,0 pontos)**

1. Imagine que você está construindo uma aplicação para um restaurante. Implemente uma classe **Produto**, com o que pode ser vendido no restaurante. A classe deverá ter como atributos a nome do produto a ser vendido, a descrição do produto e o seu preço. Por exemplo:

**Nome do Produto:** Bife ao molho madeira

**Descrição:** O prato serve quatro pessoas. Acompanha arroz branco e purê de batatas.

**Preço:** R$ 78,90

A classe **Produto** também deve ter um método void infoProduto() que imprime as informações do produto na tela.

Cada cliente antes de solicitar seus pedidos, recebe uma comanda na qual os produtos serão incluídos. A **Comanda** deve ter um número (identificador da comanda), quais produtos foram pedidos **(10 itens)** e a quantidade pedida de cada produto. Ela deve ter um método que calcula o total do pedido (double calculaTotal()). Esse método vai calcular o valor total do que foi consumido. Além desse método, a comanda deve ter um método para adicionar item void addItem(Produto p, int quantidade)e um método void infoComanda() que imprime os detalhes da comanda.

Exemplo de saída do método infoComanda():

Número da comanda: 01

Produtos:

- Bife ao molho madeira: R$ 78,90

- Refrigerante: R$ 5,00

Total da Comanda: R$ 84,90

Crie uma **classe de aplicação** que inclua 10 produtos em uma comanda, exiba as informações da comanda. **(3,0 pontos)**

1. Crie a classe **Carta**, que possui um nome e um naipe. Crie os métodos get e um construtor que configure os valores de cada carta. Crie agora uma classe **Baralho**, que possui 56 cartas. No construtor de Baralho, inicialize as 56 cartas (juntamente com os 4 coringas, que devem se chamar “coringa” e o naipe é “coringa”). Implemente os seguintes métodos **(3,0 pontos)**
   1. **embaralha( )** – usando o método Math.random dá para trocar as cartas dentro do baralho, misturando-as (sorteie duas posições e troque-as. Repita este processo diversas vezes).
   2. **daCarta( )** – devolve uma carta, retirada do topo do baralho, se não estiver vazio, ou null.
   3. **imprimeBaralho( )** – imprime as cartas para verificar como estão dispostas (se estão embaralhadas por exemplo).
   4. **temCarta( )** – verifica se tem carta no baralho, devolvendo true ou false. A classe Baralho, assim construída, será útil para programar diversos jogos de cartas, através de novas classes que os implementem.
   5. Implemente uma **classe de aplicação** que use os métodos criados.