Introdução à Programação Orientada a Objetos – IBM1094

Lista de Exercícios 02

Prazo de entrega: 26/09/2014

Aula 04 – Arrays em Java

1) As seguintes afirmações são verdadeiras ou falsas? Justifique as alternativas falsas.
   a. Um vetor pode armazenar diferentes tipos de variáveis
   b. O índice de acesso dos vetores deve ser do tipo float
   c. Se um vetor for passado como parâmetro a um método com tipo de retorno void e um de seus elementos for modificado, ao final do método o vetor terá seu valor modificado.
   d. Argumentos passados pela linha de comando são separados por vírgulas
   e. Para reservar espaço na memória, basta eu declarar a variável.

2) Escreva um aplicativo que que leia uma determinada quantidade de números, entre 10 e 100. A cada número lido, o aplicativo deve exibir uma mensagem caso o número entrado seja um número repetido. Ao final, o aplicativo deve exibir os números que foram lidos somente uma vez.

3) Implemente um aplicativo para controlar os candidatos de uma prova de concurso público. O aplicativo deve armazenar os nomes dos candidatos inscritos no concurso. O número máximo de candidatos armazenados deve ser informado como argumento de linha de comando. O sistema deve ter as seguintes opções:

   1 – Cadastrar candidato
   2 – Listar candidatos
   3 – Contar candidatos
   4 – Sair

   A opção 1 (cadastrar candidatos) só deve permitir que um candidato seja cadastrado por vez. Para cadastrar mais de um candidato, essa opção precisa ser usada múltiplas vezes.

4) Implemente um aplicativo que a leitura de duas matrizes A e B e posteriormente informe o resultado da soma das mesmas. A quantidade de linhas e colunas das matrizes deve ser passada pelo usuário como um argumento da linha de comando, e o valor de cada elemento deve ser lido do teclado.

Aula 05 – Campos e Métodos Estáticos

5) Implemente um método estático que use os métodos da classe Math para encontrar o maior e menor valor de um array de 5 elementos passado como parâmetro. Escreva um aplicativo para testar este método.

6) Implemente um método estático chamado *isMultiple* que determina para um par de números inteiros se o segundo número é múltiplo do primeiro. O método deve aceitar dois argumentos inteiros e retornar **true** se o segundo for um múltiplo do primeiro e **false** caso contrário. Incorpore este método a um aplicativo que insere uma série de pares inerios (um par por vez) e determina se o segundo valor em cada par é um múltiplo do primeiro.

7) Um problema comum de uma mesa de bar é que todo mundo divide o que está sendo pedido mas ninguém sabe o valor que precisa pagar da conta na hora que vai embora: sempre tem um que chega mais tarde ou outro que sai mais cedo que o grupo, de forma que na hora de pagar há sempre a velha luta dos clientes com a conta no guardanapo de papel. Afinal, cada um quer pagar apenas sua parte do que foi consumido, não é? (Claro, adicionando-se os 10% do garçom...).

   Uma maneira de resolver esse problema é que cada pessoa mantenha a informação de quanto lhe cabe pagar de cada pedido feito enquanto está na mesa. Assim, por exemplo, caso um cliente A tenha chego numa mesa de amigos com seis pessoas e uma bebida que custa R7,00 tenha sido pedida (para ser dividida) logo em seguida, aquele cliente deve manter anotado que ele já tem em consumo R$1,00 (R$7,00 / 7 pessoas)). Assim como o cliente A, os outros clientes estão mantendo as suas contas atualizadas. Quando um cliente quer ir embora, basta adicionar 10% ao valor guardado até então e pagar sua parte, sem maiores dificuldades.

   Crie um aplicativo Java que simule esse comportamento. O aplicativo deve possibilitar ao usuário as opções "adicionar um cliente" (cliente chega), "retirar um cliente" (cliente paga sua conta e vai embora) e "fazer um pedido", além de uma opção para sair. Ao adicionar um cliente deve ser dado um nome a ele para facilmente identificá-lo. Ao fazer um pedido, o usuário deve indicar o valor total do pedido. Após cada ação do usuário, deve ser apresentado o valor acumulado de cada cliente presente na mesa.

   Para este exercício, complete e implemente a classe Cliente abaixo com os atributos e métodos necessários. Nota: a quantidade de clientes na mesa deve ser guardada como um atributo estático da classe cliente!

```
public class Cliente {

	/**
	 * Construtor padrão. Garante que todos os clientes saibam que mais uma
	 * pessoa chegou na mesa.
	 */
	public Cliente() {...}

	/**
	 *  Método que atualiza o nome do cliente.
	 */
	public void setNome(String nome){...}

	/**
	 * Método que retorna o nome do cliente.
	*/
	public String getNome() {...}

	/**
	 * Método que notifica ao cliente que a mesa pediu um novo prato ou bebida
	 * de um dado valor. O cliente deve calcular a parte que lhe cabe em relação à
	 * quantidade de pessoas na mesa e atualizar-se de acordo.
	 */
	public void novaDespesa(float valor) {...}

	/**
	 * Método que retorna o valor atual de sua conta.
	 * O método deve incluir neste cálculo os 10% do garçom.
	 */
	public float getConta() {...}

	/**
	 * Método para fechar a conta de um cliente. Deve garantir que todos os
	 * outros clientes saibam que a partir de agora há uma pessoa menos na mesa.
	 * O cliente que chama este método não deve mais fazer parte do aplicativo.
	 * Este método deve ser usado quando o cliente sai da mesa.
	 */
	public float pagarConta() {...}
}
```