PROGRAMAÇÃO II – LISTA DE EXERCÍCIOS 8

1. Crie uma classe chamada *ItemNotaFiscal* que deverá conter os seguintes atributos:

- O número do item faturado (um número inteiro).
- A descrição do item.
- A quantidade comprada do item (um valor inteiro).
- O preço unitário do item.

Acrescente os seguintes métodos à classe:

- Um método chamado *valorTotal* que retorna o valor total do item (multiplicando o valor total pela quantidade).
- Um método chamado *valorDescontado* que recebe como parâmetro um valor *double* contendo o percentual de desconto a ser aplicado ao item e calcula o valor total descontado, seguindo a seguinte expressão:

$$VALORTOTAL \times \left(1 - \frac{DESCONTO}{100}\right)$$

Utilize o conceito de encapsulamento nesta classe. Crie um construtor para a classe que receba quatro parâmetros e permita iniciar os atributos do objeto com os valores passados. Crie outro construtor que não receba parâmetros e inicialize o objeto com valor zero para quantidade e preço, um string vazio para a descrição e o valor zero para o número do item. Faça um programa que crie um objeto da classe com valores informados pelo usuário e mostre os valores contidos nos atributos, o valor total do item e o valor total descontado de 15% (utilizando o método *valorDescontado*).

2. Implemente dois construtores para a classe *Data* mostrada abaixo. O primeiro construtor não deve receber parâmetros e deve preencher os atributos do objeto com a data do sistema (use o método *preencheDataAtual*). O segundo construtor deve receber três parâmetros inteiros e atribuir o valor de cada um a um atributo diferente (este construtor servirá para iniciar o objeto com uma data específica). Se a data armazenada for inválida, os atributos devem receber o valor 1.

```
import java.util.*;

public class Data {
	private int dia;
	private int mes;
	private int ano;

	private void preencheDataAtual() {
		GregorianCalendar cal = new GregorianCalendar();
		dia = cal.get(Calendar.DAY_OF_MONTH);
		mes = cal.get(Calendar.MONTH) + 1;
		ano = cal.get(Calendar.YEAR);
	}

	// Retorna a data como um string
	public String paraString(){
		return String.format("%02d/%02d/%04d", dia, mes, ano);
	}
}
```

3. Escreva uma classe chamada *Nave*. A nave tem direção (Norte, Sul, Leste e Oeste), velocidade, peso e altura do solo).

a. Criar um construtor que inicialize todos os atributos da Nave ao criá-la;
b. Crie um construtor que inicialize apenas o peso. Os outros dados serão direção = NORTE, velocidade = 0; altura = 0;
c. Crie um construtor sem parâmetros onde os dados serão
   - Direção = NORTE, velocidade = 0; altura = 0; peso = 100;
d. Crie métodos para ler cada um dos atributos (getVelocidade, getPeso ,etc) e crie métodos para ajustar os valores (setDireção apenas) .
e. Crie métodos de controle chamado aumentaVelocidade e diminuiVelocidade que aumenta e diminui a velocidade de 100 em 100 km por hora. Não pode haver velocidade negativa e nem pode haver velocidade acima dos 20000 km por hora.
f. Crie métodos para subir e descer, que aumenta de 100 em 100 metros de altura e decresce de 100 em 100 metros. A altura não pode ser menor que 0.
g. Crie um método Status para mostrar todos os atributos da nave, no formato abaixo:
   - Velocidade 1000 km/h
   - Altura: 45000 metros
   - Direção: Leste
   - Peso: 10000 kg

Crie um programa que permita ao usuário controlar uma nave. A nave inicialmente deve ter velocidade = 0, altura = 0, estar direcionada para o norte e ter peso de 20000 kg. Permita ao usuário que possa fazer com que a nave suba, desça, acelere, diminua a velocidade ou mude a direção da nave. O usuário poderá fazer essas operações até digitar zero para sair do programa. A cada operação feita, o status da nave deverá ser mostrado.

4. Foi solicitado por uma empresa a um programador a criação de um programa para fazer a matrícula do funcionário pelo seu nome e número de matrícula.
Abaixo está apresentado um trecho desse programa:

```
1 public class Funcionario {
2     private int matricula;
3     private String name;
4
5     public Funcionario() {
6           name = "";
7           matricula = 0;
8     }
9
10    public Funcionario(String nome, int matri){
11          name = nome;
12          matricula = matri;
13    }
14
15    public String getFuncionario(){
16          return name;
17    }
18 }
```

Em relação ao trecho de programa acima, analise as afirmativas a seguir:

I. A classe possui dois construtores: um construtor sem parâmetros, um construtor com dois parâmetros (um *String* e outro *int*) .

II. A classe possui dois atributos chamados *name* e *matricula*, que só podem ser acessados diretamente no código dos métodos da própria classe *Funcionario*.

III. A implementação do método *set* para o atributo *matricula* poderia ser:

```
public void setMatricula(int matricula) {
      matricula = matricula;
}
```

IV. Para criar um objeto da classe *Funcionario* que contenha o nome “Fulano” e a matrícula 123 e guardar sua referência em uma variável chamada *func*, poderíamos escrever:

```
Funcionario func = new Funcionario(“Fulano”, 123);
```

Está(ão) correta(s):

A. Apenas I, II e III.
B. Apenas I, II e IV.
C. Apenas I, III e IV.
D. Apenas II, III e IV.
E. Todas as afirmações estão corretas.

5. Considere a classe abaixo:

```
public class ClasseA {
      private int numero;
      public String nome;

      public ClasseA(int numero, String nome) {
            setNumero(numero);
            setNome(nome);
      }

      public void setNumero(int numero) {
            if(numero >= 0)
                  this.numero = numero;
            else
                  this.numero = 0;
      }

      public int getNumero() {
            return numero;
      }

      public void setNome(String nome) {
            if(nome == null || nome.equals(“”))
                  this.nome = “Indefinido”;
            else
                  this.nome = nome;
      }

      public String getNome() {
            return nome;
      }
}
```

Assinale a alternativa correta:

A) Caso seja criado um objeto da classe *ClasseA*, o atributo *nome* deste objeto não poderá, de nenhuma maneira, receber um *String* vazio. Caso tente-se colocar neste atributo um *String* vazio, ele receberá *"Indefinido"*.

B) Os objetos criados a partir da classe *ClasseA* não serão encapsulados.

C) O atributo *numero* pode ser alterado via o método *setNumero*.

D) O atributo *nome* só pode ser acessado diretamente em classes que estejam no mesmo pacote da classe *ClasseA*.

E) Caso se passe um valor negativo como parâmetro em uma chamada do método *setNumero*, ocorrerá um erro de execução que fará com que o programa seja encerrado.

6. Com relação à programação orientada a objetos, indique qual das afirmações abaixo está correta:

A) Construtores não podem retornar valores.

B) Para encapsular os dados de uma classe é necessário declarar todos os seus atributos com o modificador de acesso menos restritivo (geralmente o modificador público).

C) Os atributos e os métodos de um objeto especificam o seu estado em um determinado instante de tempo.

D) Os atributos de um objeto só podem armazenar dados de tipos simples, como inteiros, literais, reais ou lógicos. Uma classe não pode ser um tipo de um atributo.

E) Classes são instâncias de objetos.